UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dominik Lebar

Avtentikacija s pomočjo ponudnikov družabnih omrežij

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Ljubljana, 2014

UNIVERZA V LJUBLJANI

FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Dominik Lebar

Avtentikacija s pomočjo ponudnikov družabnih omrežij

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

MENTOR: doc. dr. Mojca Ciglarič
SOMENTOR: prof. dr. Neža Mramor Kosta

Ljubljana, 2014

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge: Avtentikacija s pomočjo ponudnikov družabnih omrežij

Preglejte področje klasičnih in sodobnih načinov avtentikacije. Pojasnite težave pri centralnih imenikih, ponudnikih identitet in pri celovitem upravljanju identitet. Opišite in primerjajte pomembnejše avtentikacijske protokole. Pojasnite pojem enotne prijave (ang. Single Sign-On) in ga primerjajte z avtentikacijo pri enem samem ponudniku. V zadnjem času se vedno bolj uporablja tudi avtentikacija s pomočjo ponudnikov družabnih omrežij. Pojasnite mehanizem takšnega prijavljanja in izpostavite prednosti in slabosti. Izberite dva izmed velikih ponudnikov družabnih omrežij in na primeru resnične spletne aplikacije implementirajte avtentikacijo prek njiju. Kritično ovrednotite ugotovitve in izkušnje, ki jih boste pri tem pridobili.

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Spodaj podpisani, Dominik Lebar, z vpisno številko **24930511**, sem avtor diplomskega dela z naslovom:

Avtentikacija s pomočjo ponudnikov družabnih omrežij.

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mojce Ciglarič in somentorstvom prof. dr. Neže Mramor Kosta,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 20. septembra 2014 Podpis avtorja:

Zahvaljujem se svoji Špeli, ker me je vztrajno podpirala pri študiju, očetu in materi, ker sta mi omogočila brezskrbno izobraževanje, bratu Luku, ker me je vsa leta spodbujal, in otrokoma Maju ter Svitu, ker sta se pridno igrala, ko sem se učil. Iskrena hvala tudi mentorici in somentorici pri nudenju pomoči pri izdelavi diplomske naloge ter Slavici Mohorč za lektoriranje.

Kazalo

Seznam uporabljenih kratic

Kratica	angleško	Slovensko
ACTIVEX	software framework created by Microsoft	programski vmesnik – avtor Microsoft
AES	advanced encryption standard	standard za napredno šifriranje po standardu FIPS PUB 197 [24]
AJAX	asynchronous JavaScript with XML	podpora za asinhrono izvajanje spletnih poizvedb in uporabo oblike XML za prenos podatkov
BASE64	encode and decode strings specified in RFC 2045	kodiranje in dekodiranje po standardu RFC 2045 [26]
CANVAS	element used to draw graphics, on the fly, with JavaScript (HTML5 standard)	risalna površina, ki je del standarda HTML5
CLOB	a character large object	tip podatka za shranjevanje večje količine podatkov v eno polje v znakovni predstavi
DOM	document object model	programski vmesnik za veljavno spletno kodo in veljavne dokumente XML
GPS	global positioning system is a satellite-based navigation system	sistem za navigacijo
HMAC-SHA1	hash based message authentication code encryption using the SHA1 hash function.	algoritem za šifriranje s funkcijo hash SHA1
HTML5	core technology markup	označevalni jezik interneta, ki se

	language of the Internet used for structuring and presenting content for the World Wide Web	uporabljaza strukturiranje in predstavitev vsebine za svetovni splet
HTTP	hypertext transfer protocol	protokol za internetni prenos spletnih strani
HTTPS	hypertext transfer protocol secure	varna, zavarovana različica HTTP
IdM	Identity management	upravljanje z identitetami
IP	address is a numerical label assigned to each device participating in a computer network	kombinacija štirih številk, ločenih s piko, pod katero se predstavlja računalnik v omrežju
JQUERY	fast, small, and feature-rich JavaScript library	knjižnica za lažje delo s programskim jezikom JavaScript
JSON	object notation using JavaScript	notacija objektov po standardu ECMA-262 [9]
MD5	widely used cryptographic hash function	šifrirni algoritem, ki uporablja zgoščevalno funkcijo
MIME	multipurpose internet mail extensions	večnamenski standard za podporo izmenjave podatkov
OAuth	open standard to authorization	odprt standard za izvedbo avtentikacijein avtorizacije z izmenjavo žetonov
OpenID	open standard and decentralized protocol by the non-profit OpenID Foundation that allows users to be authenticated by certain co-operating sites	odprt standard za izvedbo avtentikacije in avtorizacije, ki omogoča uporabo ene digitalne identitete za dostop do različnih ali nepovezanih spletnih strani
PHP	hypertext preprocessor	način kodiranja spletnih strani, programski jezik na strani strežnika

PLSQL	procedural language/structured query language	programski jezik, uporabljen v podatkovni bazi Oracle
RBAC	rolebased access control	nadzor nad dostopom z uporabo vlog
SAML	security assertion markup language	standard za avtentikacijo in avtorizacijo med različnimi domenami
SHA1	cryptographic hash function	funkcija za šifriranje hash
SOAP	simple object access protocol	protokol za izmenjavo podatkov v strukturirani obliki pri uporabi spletnih storitev
SQL	structured query language	strukturiran jezik za povpraševanje
SSL	secure sockets layer	standardza vzpostavljanje šifriranih povezav med spletnim strežnikom in brskalnikom
SSO	single sign on	sistem za enotno prijavo
TDE	Oracle transparent data encryption	šifriranje na nivoju podatkovnega prostora
TLS	transport layer protocol based on SSL	protokol za prenos podatkov na varen način z uporabo SSL
UNZIP	decompress (a compressed file or data)	razširjanje stisnjenih datotek ali podatkov (obratna operacija je stiskanje)
UTF8	universal character set	univerzalni nabor znakov
WS-TRUST	specification and OASIS standard that provides extensions to WS-Security	razširitev standarda WS-Security, ki se uporablja za varno izmenjavo podatkov v spletnih storitvah
WS-SECURITY	web services security	razširitev SOAP standarda za izvedbo varnosti v spletnih

		storitvah
XML	extensible markup language	preprost računalniški jezik, podoben jeziku HTML, za opisovanje strukturiranih podatkov
XMLHTTPREQUEST	interface available to web browser scripting languages	vmesnik, vgrajen v večino brskalnikov, za izvedbo asinhronih zahtevkov
XMLTYPE	system-defined opaque type for handling XML data in Oracle database	tip podatka za shranjevanje dokumentov XML v podatkovni bazi Oracle
XSD	XML schema, published as a W3C recommendation	shema, preko katere preverjamo veljavnost dokumentov XML
ZIP	compress file or data	stiskanje datotek ali podatkov brez izgube vsebine

Povzetek

V diplomski nalogi so predstavljene tehnologije, protokoli, oblike in postopki pri upravljanju z uporabniškimi identitetami, s katerimi moramo podpreti spletne aplikacije, da lahko uporabimo storitve družabnih omrežij. Pri spletni aplikaciji s podporo storitvam družabnim omrežjem lahko prepustimo upravljanje z identitetami zunanjemu ponudniku. Razvita je bila tudi spletna aplikacija, ki uporablja knjižnice v programskem jeziku JavaScript na strani odjemalca in programski jezik PHP na strani strežnika za vključitev podpore družabnim omrežjem in sicer z uporabo protokola Facebook Connect in protokola OAuth. Prikazan je tudi uporabljen način varnega shranjevanja strukturiranih podatkov v podatkovnem strežniku in podatkovni model, ki ga spletna aplikacija uporablja za dinamično nudenje spletne vsebine.

Ključne besede:upravljanje, identiteta, dostop, pravice, zaščita, varnost, družabno omrežje, OAuth, Facebook Connect, PHP, JavaScript

Abstract

The graduate thesis covers technologies, protocols, format of data being transfered, identity management and processes involved in the implementation of web applications supporting social network providers. It covers issues such as how users gain an identity in web applications, the protection of that identity and the technologies supporting that protection. When implementing such web applications we can use service providers for user identity management and in that case we have to use protocols needed to identify the user in a safe way. We have developed the web application using JavaScript language on the client side and PHP on the server side and covers authentication protocols including Facebook Connect and OAuth. Web application use database server for storing data in a secure and structured way. The application is using database for servicing dynamic web content.

Keywords: Identity management, identity, access, rights, protection, security, social network, OAuth, Facebook Connect, PHP, JavaScript

Poglavje 1 Uvod

Organizacije danes porabijo preveč časa za upravljanje uporabniških imen in gesel ter pravic uporabnikov. Z bolj učinkovitim upravljanjem uporabniških identitet lahko znižamo stroške za upravljanje in zvišamo raven varnosti. Zato danes učinkovito upravljanje z uporabniškimi identititeami predstavlja pomemben proces v organizacijah. Pri tem se organizacije srečujejo s problemom pri nadzoru nad dostopi do informacijskih sistemov in drugih virov, ki niso nujno informacijske narave (dostopi do prostorov organizacije, strojev v podjetju, strežnikov, računalniških delov kot so trdi diski, namiznih aplikacij, računalniških procesov, dokumentov, ipd.). Pomembo je, da proces upravljanja uporabniških identitet v vsakem trenutku odgovori na vprašanje, kdo ima dostop do kakšnega vira in ali je do njega tudi upravičen [2]. Organizacije se danes srečujejo z mnogimi izzivi, ki so si med seboj nasprotni. Na eni strani morajo nuditi dostop do podatkov zunanjim uporabnikom (npr. spletne strani, spletne storitve, dostop do podatkovnih strežnikov, ipd.), na drugi strani pa zagotavljati povečano varnost in skladnost z novimi predpisi in regulativo.

V nadaljevanju poglavje opisuje problem in motivacijo. Poglavje 2 opisuje tehnologije, ki jih danes srečamo v procesu upravljanja z identitetami. V poglavju se bomo osredotočili na postopke avtentikacije in protokole, ki nastopajo pri tem. Na kratko bomo omenili tudi oblike podatkov, ki se uporabljajo pri prenosu podatkov. Poglavje 3 prikazuje načrt rešitve, ki smo ga uporabili pri razvoju spletne aplikacije. Poglavje 4 opisuje razvoj in spletne aplikacije, tehnične podrobnosti in vanjo vključene načine zagotavljanja varnosti.

1.1 Splošno o izrazu »upravljanje z identitetami«

1.1.1 Entiteta

Gre za filozofski pojem, ki predstavlja obstojnost česarkoli v določenem prostoru in času. Pojem ni nujno vezan na materialno pojavnost. Gre za nekaj, kar je, kar obstaja. Beseda se uporablja na velikih področjih in z ustreznimi pridevniki tvori smiseln pomen. Navadno entiteto obravnavamo kot neko smiselno zaključeno enoto. Lahko je sestavljena iz več podenot.

1.1.2 Identiteta

Če ponovno pristopimo najprej filozofsko in povežemo pojem s predhodnim poglavjem, potem lahko rečemo, da identiteta predstavlja vse, kar entiteto naredi prepoznavno in unikatno določljivo. V naravoslovni znanosti bi se lahko strožje opredelili in rekli, da je identiteta vsaka podmnožica vrednosti atributov posameznika, ki enolično določa tega posameznika v katerikoli množici, v kateri posameznik nastopa. Tako ne moremo govoriti za posameznika, da zanj obstaja enolična identiteta, pač pa množica identitet[36].

1.1.3 Upravljanje z identitetami

Tehnologije, storitve in terminologija povezana z upravljanjem z identitetami vključuje storitve uporabniških imenikov, ponudnike storitev, ponudnike identitete, spletne storitve, nadzor nad dostopom do podatkov, digitalne identitete, upravljalce gesel, enotne prijave, varnostne žetone, varnostne storitve žetonov, poteke dela, ter tehnologije, protokole in specifikacije, kot na primer OpenID, WS-Security, WS-Trust, SAML 2.0, OAuth in RBAC, ki jih bomo podrobneje opisali v poglavju 2.2. Postopek upravljanja z identitetami zajema vprašanja, kot so, kako uporabniki dobijo identitete, zaščito te identitete in tehnologije, ki podpirajo to zaščito (npr. omrežni protokoli, digitalna potrdila, gesla ipd.)[49].

Sistem upravljanja z identitetami predstavlja danes v organizacijah pomembno vlogo, saj z njim zagotavljamo izboljšano dostopnost in storitve, povečamo varnost in znižamo stroške v organizaciji. Upravljanja z identitetami (Identity management) v računalništvu opisuje sistem za upravljanje posameznih uporabnikov, preverjanje njihove pristnosti, dovoljenj in privilegijev v okviru posameznega sistema ali izven njegovih meja. Pri tem je cilj sistema zagotavljanje večje varnosti, večje produktivnosti, zmanjševanje stroškov in izpadov ter zmanjševanje ponavljajočih se opravil. Sistemi za upravljanje identitete, izdelki in aplikacije upravljajo s podatki identitet in vključujejo posameznike, strojno ter programsko opremo. Poslovni razlogi oziroma cilji za uvedbo upravljanja identitet so[20]:

- znižanje stroškov:
  - znižanje stroškov upravljanja (oskrbovanje, avtomatizacija procesov, delegirana administracija in samooskrbovanje uporabnikov),
  - znižanje stroškov lastništva in hitrejši čas implementacije,
  - znižanje stroškov razvoja in integracije;
- povečanje varnosti in minimiziranje tveganja:

  - z avtomatiziranim odkrivanjem nevarnosti in poročanjem,
  - z nadzorovanim dostopom (glede na vloge in pravila),
  - s celovito revizijo posamezne identitete,
  - s proaktivno avtomatizirano in dokazljivo skladnostjo s predpisi in regulativo,
  - s shranjevanjem podatkov na varen način;
- izboljšanje dostopnosti in s tem kakovosti storitev:
  - s poenostavljenim načinom prijave z enotno prijavo (Single sign on),
  - z možnostjo ponastavitev gesel,
  - z avtomatiziranim oskrbovanjem uporabnikov,
  - z nudenjem storitev partnerjem in strankam;
- avtomatizacija procesov;
- zagotavljanje skladnosti s predpisi in regulativo z uporabo standardnih protokolov;
- omogočanje enostavnega razvoja novih poslovnih modelov;
- upravljanje in sledljivost identitet;
- podpora družabnim omrežjem.

Poglejmo si še nekaj statističnih podatkov, ki zgovorno prikazujejo, zakaj je upravljanje z identitetami[20] tako pomembno:

- povprečni uporabnik na dan porabi 16 minut časa za avtentikacijo in aktivnosti prijave v različne sisteme (Vir: Meta Group, pod Gartner Group od leta 2005),
- redni uporabniki imajo v povprečju 21 gesel (Vir: NTA Monitor Password Survey),
- 49% uporabnikov ima gesla nekje zapisana, 67% uporabnikov jih zelo redko ali celo nikoli ne zamenja,

- letno znaša izguba zaradi prevar 12 bilijonov USD (Vir: Comm Fraud Control Association),
- tipična organizacija z 2.500 uporabniki lahko letno porabi 850.000 USD za interno podporo uporabnikom (Vir: Gartner Group),
- ravnanje z gesli je ena izmed glavnih desetih varnostnih tveganj za podjetja (Vir: Gartner Group),
- stroški podpore za pomoč uporabnikom znašajo v povprečju 30 USD na klic,
- klici povezani s problematiko, ki se nanaša na dostope in gesla, predstavljajo 40% vseh internih klicev (Vir: Gartner Group).

Poglavje 2 bolj podrobno opisuje področja, na katera lahko razdelimo proces upravljanja z identitetami:

- uporabniški imenik,
- avtentikacija,
- nadzor dostopa,
- upravljanje z uporabniki.

1.2 Opis problema

V podjetju smo podprli poslovne procese s spletnimi aplikacijami, ki od uporabnika zahtevajo avtentikacijo. Z vključitvijo družabnih omrežij v spletne aplikacije smo želeli doseči večjo povezljivost med posameznimi skupinami v podjetju in uporabiti storitev upravljanja z identitetami družabnega omrežja za izvedbo avtentikacije. Pri tem se nismo želeli omejevati na posamezna družabna omrežja, pač pa smo želeli doseči čim večjo povezljivost tudi med samimi omrežji. Preučili smo možnosti, kako lahko učinkovito upravljamo uporabniške identitete s storitvami družabnih omrežij in s tem zagotovimo nadzor nad dostopi do različnih virov. Pri vsem tem smo želeli ohraniti vse vidike varnosti.

1.2.1 Kratek uvod v problem

Pojavi se vprašanje, kako lahko vzporedno učinkovito upravljamo uporabniške identitete in njihove pravice dostopov do virov organizacije in virov družabnih omrežij. Danes ima

posameznik problem, saj tipično uporablja številne različne identitete pri vključevanju v notranja in družabna omrežja. Poleg tega družabna omrežja omogočajo uporabo lažnih identitet, zato je izpostavljen tudi varnostni vidik. Vprašamo se, ali je možen prehod na eno samo identiteto. Ugotovili smo, da to ni vedno mogoče, čeprav se temu lahko navidezno približamo in uporabniku ponudimo dokaj nevsivljiv način enotne avtentikacije. Z uporabo pravih protokolov lahko zagotovimo varnost pri izvedbi avtentikacije. Kaj pa prenos podatkov? Tu varnost ni vedno zagotovljena, zato je uporaba pravih protokolov na nivoju komunikacije pomembna. Prav tako bomo izpostavili problem pri shranjevanju podatkov in poglavje 4.4 bo razkrilo eno izmed možnosti varnega shranjevanja podatkov na disku in tudi varne izmenjave podatkov, kadar nam protokoli tega ne zagotavljajo.

1.2.2 Identitete v družabnih omrežjih

Pri vključevanju družabnih omrežij v spletne aplikacije potrebujemo enega izmed načinov avtentikacije, ki ga danes ponujajo družabna omrežja. Tu gre za komunikacijo izven domene podjetja, zato enotni način prijave (single sign on) ni primeren. V tem primeru smo primorani uporabiti druge protokole in ponudnike za upravljanje z identitetami, ki omogočajo izvedbo avtentikacije na varen način. Večina družabnih omrežij danes sicer že zahteva SSL/TLS protokol, ki je že varen v smislu same povezave. Vendar določena družabna omrežja nudijo še vedno možnost komunikacije po nezaščitenem protokolu HTTP. Takrat je protokol, ki se uporablja za izvedbo avtentikacije, tudi temu prilagojen. Tipičen primer je protokol OAuth. Različica 1.0 se uporablja tudi na nezaščitenih protokolih, med tem ko je različica 2.0 poenostavljena ravno zaradi predvidene uporabe protokola SSL/TLS.

1.2.3 Pristopi k reševanju problema

V osnovi je treba vedeti, kaj želimo od spletne strani v povezavi z družabnim omrežjem. Ali želimo samo podpreti avtentikacijo z digitalno identiteto družabnega omrežja ali celo nadomestiti način avtentikacije? Morda želimo dostopati le do zaščitenih informacij družabnega omrežja, za katere je zahtevana avtentikacija ali pa dodati storitve, ki jih družabna omrežja omogočajo. Vedeti moramo, ali bomo dovolili več identitet in ali je uporaba družabnega omrežja obvezna ali ne. Vse to vpliva na način vgradnje in posebnosti moramo podpreti. Če je uporaba digitalne identitete družabnega omrežja obvezna, potem lahko uporabimo digitalno identiteto družabnega omrežja za enolično določitev posameznika in drugačne avtentikacije ne potrebujemo. Včasih v določene spletne aplikacije ne moremo posegati, zato je pomembna povezljivost z obstoječimi načini avtentikacije, ki jih organizacija izvaja. Pri vsem tem igrajo pomembno vlogo piškotki, ki pa imajo zaradi varnosti omejitve pri uporabi.

Po eni strani imamo omejitve pri shranjevanju piškotkov, ker imamo komunikacijo, ki gre izven domene podjetja, po drugi strani bi radi ohranili enotni sistem prijave (single sign on) in ga razširili s storitvami ponudnikov družabnih omrežij. Možnosti imamo več. Uporabili smo tudi več rešitev, glede na zahteve za vključitev v delovni proces. Včasih je možno programsko kodo prilagoditi, drugič te možnosti nimamo in moramo izbrati drugačen pristop. Ena izmed rešitev uporablja enotni sistem prijave, med tem ko je v posamezno spletno stran znotraj domene dodana podpora za izvedbo avtentikacije preko OAuth ali OpenID protokola. Prijavo v sistem uporabnik praviloma stori samo enkrat, potem se piškotki zapišejo tako na strani odjemalca, kot tudi v obliki strukturiranega zapisa na strežnik. Namesto da protokol obravnava identiteto z uporabniškim imenom in geslom, se ta po enkratni izvedbi avtentikacije zamenja za žeton. Zaščitene dele spletne strani lahko pridobimo z uporabo žetona, ki nadomešča identiteto. Preverjanje identitete je z uporabo protokola enostavno, prav tako tudi ravnanje z žetonom in s tem vključevanje storitev, ki jih nudijo ponudniki družabnih omrežij.

Ostaja vprašanje, kakšni so torej drugi pristopi. Spletni aplikaciji, ki je ne smemo prilagoditi, lahko spremenimo funkcionalnost in preusmerimo njeno uporabo na način, da jo uporablja izključno strežnik. Preko nove spletne strani odjemalec izvede zahtevo, strežnik zahtevo preusmeri na staro spletno stran in pridobi odgovor ter preko varnih protokolov posreduje rezultat odjemalcu. Strežnik je v tem primeru v vlogi odjemalca. S tem se izognemo potrebi po spremembi spletne aplikacije, varnejši izvedbi in uporabi varnih protokolov. Prilagoditev na uporabo varnih protokolov je brez spremembe programske kode spletne aplikacije praktično nemogoča. Pri vključitvi spletne aplikacije stranke, kateri arhitekture ne smemo spreminjati, se s skupno dopolnitvijo aplikacijo prilagodili tako, da tudi brez uporabe varnih protokolov dosežemo večji nivo varnosti. Če je spletna aplikacija gostujoča na strežniku ponudnika storitve, želimo narediti spremembe v kodi na obeh straneh čim manjše in čim bolj enostavne.

1.3 Motivacija

Pri uporabniških identitetah in njihovih pravicah dostopov do podatkov ne gre samo za zapise v uporabniškem imeniku, kjer se največkrat vodijo uporabniške lastnosti, temveč tudi za druge sisteme, aplikacije, spletne storitve, družabna omrežja in druge informacijske vire. Informacijskih virov, do katerih morajo imeti uporabniki pooblastilo za uporabo, je vedno več. Vedno več je tudi zunanjih uporabnikov, ki uporabljajo informacijske vire podjetja. Pri tem je največji pomislek varnost. Težava današnjih omrežij je, da po večini vsako zahteva svojo identiteto. Tako ima uporabnik problem s številnimi identitetami. Po drugi strani tudi

omogočajo uporabnikom, da imajo lažne identitete. Vse to lahko rešimo s prehodom na eno samo identiteto, kot je npr. enotna prijava (single sign on), vendar to ni vedno mogoče. Želeli bi, da je dodeljevanje, spreminjanje in brisanje pooblastil natančno urejeno, nadzorovano in zaradi veliko sprememb tudi v največji možni meri avtomatizirano. Z vključitvijo zunanjih ponudnikov upravljanja z identitetami dodamo podporo za družabna omrežja z uporabo standardnih protokolov za namen zagotavljanja varnosti.

Osnovna ideja pri nastanku rešitve je bila izdelava vmesnika za izdelavo spletnih strani z enostavno možnostjo vključitve družabnih omrežij in storitev, ki jih ponujajo. Vgradnja spletne aplikacije v sistem aplikacij družabnih omrežij danes v večini primerov zahteva varni način komunikacije po protokolu HTTPS in izvedbo prijave v družabno omrežje po protokolu OAuth, ki je podrobno opisan v diplomskem delu z naslovom »Odprta avtentikacija (Open authentication)«[10]. V realnem svetu pravzaprav največjo težavo predstavljajo obstoječe spletne aplikacije, v katere ne moremo bistveno posegati oziroma sploh ne moremo. Preverili smo, kakšne možnosti imamo v tem primeru. Kaj pa, kadar spletna aplikacija ne uporablja varnih protokolov za izmenjavo podatkov? Kakšen pristop moramo takrat uporabiti? Z izdelavo vmesnika smo pridobiti naslednje prednosti:

- hiter razvoj spletnih aplikacij povezanih z družabnimi omrežji,
- povezava v več družabnih omrežij hkrati,
- zagotovitev možne integracije z obstoječimi spletnimi aplikacijami, ki nimajo vgrajene podpore za uporabo varnih protokolov,
- metodo za vključitev spletnih aplikacij na nivo varnega komuniciranja, katerih programske kode ne moremo ali ne smemo spreminjati,
- vključitev v interne procese znotraj organizacije.

Poglavje 2 Pregled pojmov in tehnologij

V poglavju si bomo najprej ogledali nekaj splošnih informacij, funkcije in področja pri upravljanju z identitetami, ki v grobem zajemajo uporabniški imenik, avtentikacijo, avtorizacijo in kontrolo dostopa. Sledil bo poudarek na sami avtentikaciji, kjer si bomo pogledali različne protokole in dopolnitve k protokolom za izvedbo avtentikacije. V poglavju bomo potem prikazali še načine prijave in zunanje ponudnike upravljanja z identitetami, kot so OpenID, OpenID Connect, OAuth, Facebook Connect, Google account.

2.1 Upravljanje z identitetami

V poglavju si bomo ogledali nekaj splošnih informacij, funkcije in področja pri upravljanju z identitetami.

2.1.1 Funkcije upravljanja identitete

V kontekstu izgradnje računalniških sistemov ima upravljanje z identitetami tri osnovne funkcije[49]:

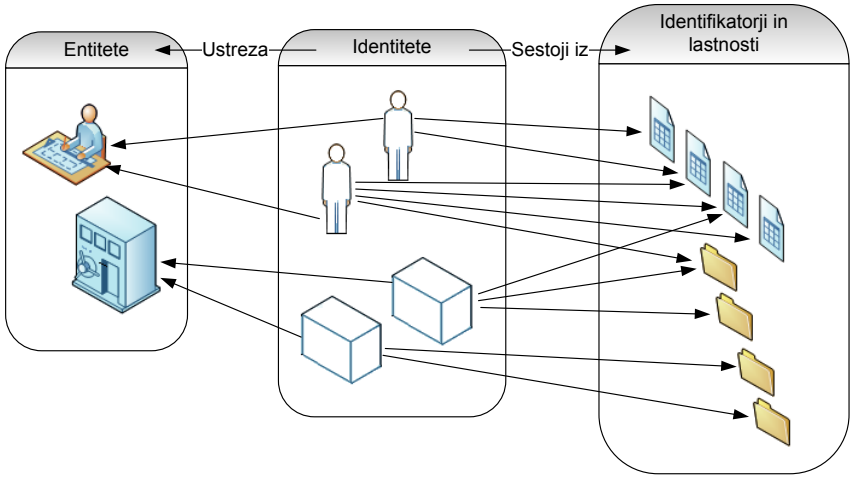
- funkcija čiste identitete: kreiranje, upravljanje in brisanje identitet ne glede na pravice dostopa,
- funkcija dostopa uporabnika (prijava): na primer z uporabo pametne kartice in podatki, ki jih vnese stranka, da se prijavi v sistem storitev,
- funkcija storitev: sistemi, ki prinašajo uporabniške nastavitve, pravice z vlogami, spremembe v živo ali na zahtevo, večpredstavnostno vsebino in nudenje teh storitev uporabnikom ter napravam.

V nadaljevanju si poglejmo funkcije bolj podrobno.

2.1.1.1 Čiste identitete

Splošni model identitete je sestavljen iz majhnih množic aksiomov, kot je npr., da so vse identitete v določenem prostoru in času enolično določene ali pa, da imajo povezavo z entiteami v resničnem svetu. Tak model predstavlja čisto identiteto v smislu, da model ni

omejen z določenim naborom aplikacij ali njihovim kontekstom. V splošnem ima lahko ena entiteta več identitet in vsaka identiteta sestoji iz več lastnosti, ki so lahko enolično določene glede na prostor. Slika 1 prikazuje identitete, entitete, njihove lastnosti in povezave med njimi.



Slika 1: Koncept identitete

V večini teoretičnih in vseh praktičnih modelih digitalne identitete digitalna identiteta sestoji iz končne množice lastnosti, ki imajo svoje vrednosti. Te lastnosti nosijo informacijo o objektu za namene uporabe zunaj modela ali za upravljanje modela. Na primer za razvrstitev ali iskanje. Čista identiteta se ne ukvarja z zunanjo semantiko teh lastnosti. Najpogosteje v praksi iz čiste identitete izpeljemo zahtevo za identifikacijo preko podpisa ali programskega žetona, ki ga model interno uporabi za zagotavljanje enolične identitete. Če model prestavlja to semantiko znotraj modela, se ne šteje več kot čisti model. V nasprotju s tem govorimo o lastnostih, ki so od zunaj uporabljene za namen pridobitve informacij za izvedbo varnega in upravičenega dostopa, vendar so hranjene na enostaven način, brez vmešavanja samega modela. Pomanjkanje zunanjih semantik znotraj modela kvalificira model kot model čistih identitet. Upravljanje z identitetami lahko potem definiramo kot nabor operacij na določenem modelu oz. nabor zmožnosti, ki se sklicujejo na model. V praksi je upravljanje z identitetami pogosto razširjeno na način, da zajema uporabo večih modelov za upravljanje z identitetami hkrati.

Poleg kreiranja, brisanja ali spreminjanja identitete uporabnika s tujo pomočjo ali avtomatsko, je postopek upravljanja z uporabniki zadolžen za nadzor pomožnih entitet za uporabo v aplikacijah, kot so npr. kontaktne informacije ali lokacije.

2.1.1.2 Uporabniški dostop

Uporabniški dostop (User access) omogoča uporabniku dostop do različnih aplikacij z uporabo ene digitalne identitete in kontrolo, ki se izvaja nad to identiteto. Uporaba ene same identitete za več sistemov poenostavi upravljanje s takimi identitetami in kontrolo dostopa posamezne identietete. S tem je poenostavljen nadzor in preverjanje, ter hkrati pomaga organizacijam, da zmanjšajo potrebno dodeljevanje pravic na minimalno vrednost in s tem ne dodeljujemo več pravic, kot je zahtevano. Zadoščeno je sledenju uporabnika od začetka do prekinitve uporabniškega računa. Ko organizacija uvaja proces upravljanja z identitetami v sistem, je motivacijski namen dodeljevanje primernih pravic entitetam preko njihovih identifikatorjev. Z drugimi besedami, je uporabniški dostop praviloma motivacija za uvedbo upravljanja z identitetami.

2.1.1.3 Storitve

Organizacije vključujejo vedno nove storitve tako za notranjo uporabo znotraj organizacije, kot tudi navzven za zunanje uporabnike oz. stranke. Veliko od teh storitev zahteva upravljanje z identitetami, za pravilno nudenje storitev. Upravljanje z identitetami je vse bolj ločeno od funkcij aplikacij, tako da se ena sama identiteta uporabi za eno ali največkrat kar vse aktivnosti znotraj organizacije. Za notranjo uporabo se upravljanje z identitetami razvija v smeri nadzora dostopov do digitalnih vsebin, naprav, mrežne opreme, strežnikov, portalov, vsebin dokumentov, aplikacij in/ali produktov. Storitve največkrat zahtevajo veliko informacij o uporabniku, kot so npr. naslovi, lastnosti, pravice in kotnaktne informacije. Ker je večina teh informacij zaupne narave, je ključno zagotavljanje varnosti na tem področju.

2.1.2 Zasebnost in varnost

Osebni podatki seveda odpirajo vprašanja zasebnosti. Pomanjkanje varnosti in zaščite lahko pripelje do tega, da tretji uporabnik pridobi kontrolo nad podatki in procesi organizacije. Družabna omrežja na veliko uporabljajo upravljanje z identitetami. Na kakšen način uporabnik določi, katere informacije bo razkril v omrežju, je postalo pereče vprašanje.

2.1.2.1 Kraja identitete

Do kraje identitete pride, ko nekdo pridobi identiteto, kot je na primer številka PIN, ki dovoljuje dostop do bačnih računov. Novi tipi mrežnih napadov[15] kažejo na to, da je danes varnost zelo pomembna tudi pri uporabi protokola SSL/TLS. Prenos podatkov o vrednosti žetona je veliko bolj primeren, kot če po omrežju prenašamo uporabniško ime in geslo. Na tem slonijo vmesniki za odprtokodne avtentikacije[10]. Pri tem igra enako pomembno vlogo način shranjevanja podatkov na podatkovnem strežniku. Tehnologija podjetja Oracle omogoča danes, ne samo šifriranje podatkov na datotečnem sistemu, pač pa tudi maskiranje

podatkov. Z uporabo maskiranja uporabniku ne prikažemo podatkov, ki za uporabnika niso primerni, pač pa namesto podatkov prikažemo zvezdice.

2.2 Področja pri upravljanju z identitetami

Pogledali smo si osnovne funkcije upravljanja z identitetami. V nadaljevanju si poglejmo tehnološka področja. Upravljanje z identitetami bi lahko razdelili na naslednja področja:

- uporabniški imenik;
- avtentikacija (gre za proces preverjanja entitete, in sicer se preveri, za koga ali kaj se entiteta predstavlja, preverjanje se izvede z uporabo gesel, biometrične tehnologije, kot je prstni odtis ali pa z znakom, ki ga naredimo s premiki prstov na napravah občutljivih na dotik);
- avtorizacija (avtorizacija določi kakšne operacije lahko entiteta opravlja v okviru določene aplikacije, uporabnik ima lahko npr. pravico za izpolnitev obrazca, medtem ko ima drug uporabnik lahko samo pravico za potrjevanje naročila);
- nadzor dostopa preko vlog (vloga (roles) je skupek operacij in/ali skupek drugih vlog, uporabnikom je tipično dodeljena določena vloga za opravljanje točno določene funkcije ali opravila, na primer bazni administrator lahko v okviru vloge izvede spremembo gesel uporabnikov, medtem ko sistemski administrator lahko uporabniku dodeli dostop do določenega strežnika);
- delegacija (delegacija (Delegation) dovoli lokalnim administratorjem ali nadzornikom, da naredijo sistemske spremembe brez globalnega administratorja ali da izdajo dovoljenje drugim uporabnikom, da lahko izvajajo akcije v njihovem imenu, npr. uporabnik lahko delegira pravico za upravljanje z informacijami, ki so vezane na pisarno);
- izmenjava (za izmenjavo podatkov o identitetah med različnimi domenami se uporablja protokol SAML).

V nadaljevanju so glavna področja bolj podrobno opisana.

2.2.1 Uporabniški imenik

Vsak uporabnik je v imeniku predstavljen kot objekt. Informacije o uporabniku so predstavljene kot lastnosti tega objekta. Informacije znotraj objekta so lahko zaščitene na

način, da do njih lahko dostopa samo uporabnik s potrebnimi pravicami. Bolj napredni imeniki so izdelani tako, da se delijo na imenske prostore, kot so naročniki, storitve, naprave, pravice, dolžnosti, vsebine. Načrtovanje uporabniškega imenika je močno povezano s pojmom »upravljanje z identitetami«.

Imenik definira prostor za omrežje ali uporabnike. Prostor je v našem kontekstu nekaj, kar vsebuje enega ali več objektov. Imenik je ponavadi definiran tako, da ima pravila, ki natančno določajo, kako se imenujejo podatkovne poti in kako so razpoznane. Pravila določajo, da ima vsak uporabnik enolično in nedvoumno oznako. V standardu X.500 (standard za uporabniške imenike) in LDAP tako oznako imenujemo »enolično ime« (Distinguished name) in pomeni skupek lastnosti, ki sestavljajo ime zapisa v imeniku. V bolj kompleksnih primerih je imenik centralno skladišče za nudenje storitev. Na primer, ko uporabnik išče računalnik v imeniku, dobi seznam računalnikov in informacije za dostop do računalnika.

Poseben pomen pri imenikih imata postopka replikacije in distribucije. Replikacija se uporablja za kopiranje enakih objektov v drug imenik z namenom zagotavljanja dostopnosti storitve (če se na primer en strežnik pokvari) ali z namenom zagotavljanja boljše odzivnosti in s tem storitve. Replikacijo izvaja ponavadi isti organ, kot je določen za upravljanje prvotnega imenika. Distribucija pa se uporablja, ko želimo nek imenik razbiti na ločene celote. Imeniška strežnika sta med seboj povezana, da skupaj tvorita distribuirano imeniško storitev. Ponavadi ločena prostora upravljajo tudi ločeni organi (ločene odgovorne osebe).

Med glavne aktivnosti uporabniškega imenika lahko v grobem štejemo dodajanje, brisanje in urejanje uporabniškega računa. Vendar pod pojem urejanje lahko štejemo veliko podaktivnosti, kot so:

- sprememba imena,
- dodajanje / spreminjanje gesla,
- spreminjanje slike,
- spreminjanje vrste računa,
- določanje kvote,
- določanje omejitev uporabe,
- časovne omejitve,
- starševski nadzor,

- nadzor iger.

Predstavimo nekaj tehnologij povezanih z uporabniškimi imeniki:

2.2.1.1 Protokol LDAP (Lightweight Directory Access Protocol)

Protokol LDAP (Lightweight Directory Access Protocol) je aplikacijski protokol za povpraševanje in spreminjanje objektov pri ponudniku imenika, kot je na primer Active Directory, in uporablja obliko LDAP. Pogosto uporabljena kratica AD predstavlja podatkovno zbirko imeniška storitev, LDAP pa je eden izmed protokolov, ki jih uporabimo za komunikacijo z njim.

2.2.1.2 Ponudniki uporabniških imenikov

Večina ponudnikov uporabniških imenikov se nagiba k uporabi standardov zajetih pod oznako X.500 in protokolom LDAP (2.2.1.1). Primeri izvedb uporabniških imenikov so še Red Hat 389 Directory Server (predhodno imenovan Fedora/Red Hat Directory Server), Microsoft Active Directory (2.2.1.1), Apache Directory Server, Apple Open Directory, CA Directory, Critical Path Directory Server, DirX Directory, FreeIPA, IBM Tivoli Directory Server, ldapjs (implementacija protokola LDAP v programskem jeziku JavaScript), Mandriva Directory Server (sedaj del produkta Mandriva Management Console), Nexor Directory, Novell eDirectory, OpenBSD ldapd, OpenDJ, OpenDS, OpenLDAP, Virtual Identity Server podjetja Optimal IdM, Oracle Directory Server Enterprise Edition, Oracle Unified Directory, RadiantOne, Samba4, Slapd (samostojen LDAP strežnik), Sun Java System Directory Server, UnboundID Directory Server, Univention Corporate Server in ViewDS Directory Server (platformsko neodvisen X.500/LDAP imeniški strežnik). Ponudnikov je veliko, med najbolj razširjenimi v podjetjih, ki uporabljajo programsko opremo Windows, je Microsoft Active Directory[51].

2.2.1.3 Aktivni imenik Microsoft (Microsoft Active Directory)

Aktivni imenik je skupek vseh uporabnikov in njihovih lastnosti. Aktivni imenik podjetja Microsoft je danes med najbolj razširjenimi produkti, ko govorimo o ponudnikih uporabniških imenikov in deluje v Windows okolju. Sistem uporablja podatkovno bazo, ki zagotavlja preverjanje pristnosti, imenik, politike in druge storitve v okolju Windows. Aktivni imenik je imeniška storitev, s katero pridobimo centralno avtentikacijo in avtorizacijo za vse računalnike, ki uporabljajo operacijski sistem Windows. Imenik omogoča upravljalcu dodeljevanje pravic in nastavljanje pravil, nameščanje programske opreme in nameščanje nadgradenj ter popravkov. Aktivni imenik hrani informacije v centralni, organizirani in dostopni podatkovni bazi in to ponavadi za celotno podjetje. Storitev je bila razvita s strani podjetja Microsoft in je vključena v večini strežnikov z operacijskih sistemom Windows, kot

nabor procesov in storitev. Ko se uporabnik prijavi na računalnik, ki je v domeni Windows, aktivni imenik preveri geslo in preveri, ali gre za navadnega uporabnika ali za administratorja. Aktivni imenik uporablja protokol LDAP (Lightweight Directory Access Protocol) različico 2 in 3, lastno zaščito Kerberos in strežnik DNS (Domain Name Server).

2.2.2 Avtentikacija

Ker gre za ključno poglavje v diplomskem delu, smo avtentikaciji namenili ločeno poglavje 2.3. V nadaljevanju pa si poglejmo še druge funkcionalnosti, ki so tesno povezane s procesom upravljanja z identitetami.

2.2.3 Avtorizacija

Avtorizacija je funkcija določanja pravic za dostop do virov in pravice povezane z uporabo virov. Ponavadi izraz srečujemo na področju varnosti v računalništvu, za zagotavljanje nadzora dostopa do podatkov ali storitev. Če se izrazimo bolj formalno, pomeni določiti politiko varnosti. Npr., ko uporabnik želi dostopati do vira podatkov, se mora avtenticirati in pravila določajo, kaj uporabnik lahko vidi, bere oziroma dela. Pravila določajo, ali uporabnik ima dostop do vira, njegove vsebine ali je dostop zavrnjen. Viri zajemajo lahko posamezne datoteke ali pa podatke v datotekah, programe, naprave in funkcionalnosti znotraj posameznih aplikacij. Primeri so uporabniki računalnikov, programi na računalniku in druge naprave na računalniku. Nadzor nad kontrolo dostopa v računalniških sistemih in omrežjih največkrat sloni na politiki dostopov. Postopek dodeljevanja in nadziranja pravic bi lahko razdelili na:

- opredelitev politike varnosti - kdaj je dostop dovoljen ali zavrnjen,
- izvajanje politike varnosti, kjer so zahteve za dostop do določenega vira odobrene ali zavrnjene.

Dovoljenja so torej določena v fazi opredelitve, ki je pred fazo izvrševanja kontrole, kjer so zahteve za dostop odobrene ali zavrnjene na osnovi predhodno določenih dovoljenj. Večina današnjih operacijskih sistemov vključuje kontrolo dostopov in so odvisni od avtorizacije. Kontrola dostopa uporablja avtentikacijo za preverjanje identitete uporabnika. Ko uporabnik želi dostopati do vira, kontrola dostopa preveri, ali je uporabnik avtoriziran za uporabo vira. Avtorizacija je tudi odgovornost organa, kot je npr. vodja oddelka znotraj področja uporabe, vendar pogosto se te odgovornosti prenašajo na skrbnika sistema.

Pod pojmom avtorizacija se skrivajo pravila za dostop, npr. v obliki seznama pravil in možnosti. Ponavadi so pravila zapisana na način, da uporabnik dobi najmanjši nabor

privilegijev za izvedbo opravil. Uporabnik naj bi bil avtoriziran samo za storitve ali objekte, ki jih potrebuje za izvedbo opravil, ki jih izvaja. Stari sistemi ali operacijski sistemi, ki so temeljili na enem samem uporabniku, največkrat niso imeli mehanizma kontrole dostopa in avtentikacije. Anonimni uporabniki ali gosti so stranke, za katere avtentikacija ni potrebna in imajo praviloma omejen dostop. V distribuiranih sistemih je zaželjeno, da lahko dodelimo pravice brez zahteve po enolični identiteti. Znani primeri vključujejo uporabo žetonov za dostop, ki uporabljajo ključe in omogočajo dostop brez posredovanja podatkov o sami identiteti. Strankam, ki jim zaupamo, pogosto omogočimo neomejen dostop do informacij v našem sistemu, vendar se morajo avtenticirati, da dobijo dovoljenje za izvajanje operacij, kot so dostopi do podatkov ali na primer uporaba internih spletnih storitev. Gosti in stranke, ki jim ne zaupamo popolnoma, bodo pogosto imeli omejen dostop do vsebin z namenom zaščite, pravilne uporabe ali kontroliranega dostopa do informacij. Politika dostopa v nekaterih operacijskih sistemih omogoča dostop do vseh informacij vsem uporabnikom. Drugi operacijski sistemi tega ne omogočajo in zahtevajo od administratorja, da poda dovoljenja za dostop do vsakega vira. Tudi kadar je dostop kontroliran preko nabora možnih izvedb avtentikacije in kontrol dostopa, vzdrževanje teh informacij ni enostavno, zato pogosto predstavljajo breme tako pri upravljavljanju, kot tudi pri izvajanju same avtentikacije. Velikokrat se pojavi zahteva po spreminjanju ali odstranitvi uporabnikove možnosti po avtorizaciji in to dosežemo s spreminjanjem ali brisanjem pravil v sistemu. Uporaba atomarne avtorizacije, kjer tretja oseba (ali organizacija oz. spletna stran) omogoča distribucijo informacij za izvedbo avtorizacije na varen način, je alternativa sistemski avtorizaciji.

2.2.4 Kontrola dostopa (Access control)

2.2.4.1 Kontrola dostopa s seznamom (Access control list ali ACL)

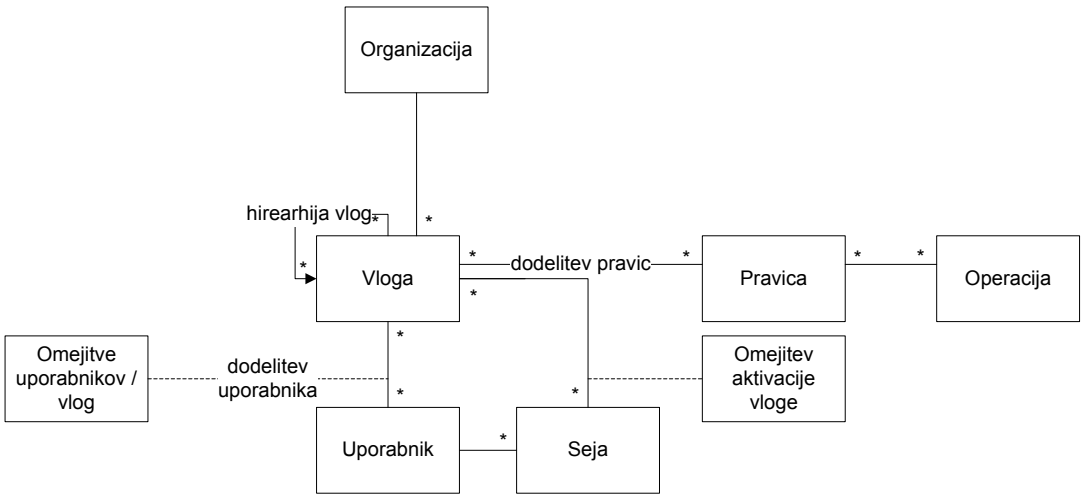
Kontrola dostopa s seznamom se uporablja na nivoju datotečnega sistema. Gre za seznam pravic, tipično v datoteki, ki se navezuje na nek objekt. Seznam pove, kateri uporabniki ali sistemski procesi imajo dovoljenje za dostop in operacije nad objekti, za katere seznam določa pravice[32]. Zapis v seznamu tipično določa objekt in operacijo. Na primer, če seznam vsebuje zapis »Janko: read,write; Metka: read«, to pomeni, da ima Janko pravico pisanja in branja, Metka pa ima samo pravico branja.

2.2.4.2 Dostop glede na vloge (Role based access control ali RBAC)

V računalniških sistemih pri opredelitvi pravic uporabnika pogosto uporabljamo vloge. Vloga vsebuje nabor pravic in/ali nabor drugih vlog, kot prikazuje Slika 2. Način omejevanje RBAC [11],[39] nudi omejevanje dostopa glede na uporabnika, ki se avtorizira. Omejevanje se preveri glede na vloge, ki so uporabniku dodeljene, in tak sistem se uporablja v večini

organizacij, ki ima več kot 500 zaposlenih[35]. S takim omejevanjem načina dostopa lahko določimo, kaj so obvezne vloge (v angleškem jeziku mandatory access control ali s kratico MAC), ki jih mora imeti uporabnik za izvedbo avtentikacije in določitev dostopa ali na primer določitev neomejenega (v angleškem jeziku discretionary access control ali s kratico DAC) dostopa do virov. Pri omejevanju dostopa na način RBAC imamo tri glavna pravila:

- dodeljevanje vlog (uporabnik lahko uveljavlja pravico samo v primeru, če je bila uporabniku dodeljena vloga, v kateri je zajeta pravica),
- avtorizacija vlog (uporabnikova aktivna vloga mora biti avtorizirana, kar v kombinaciji s prvim pravilom zagotavlja, da uporabnik lahko dobi samo vloge, za katere je avtoriziran),
- avtorizacija pravice (uporabnik lahko uveljavlja pravico samo v primeru, če je pravica avtorizirana v okviru uporabnikove aktivne vloge, kar v kombinaciji s prvim in drugim pravilom zagotavlja, da uporabnik lahko uveljavlja samo pravice, za katere je avtoriziran).



Slika 2: Shematski prikaz dostopa glede na vloge (RBAC)

Uporaba RBAC za upravljanje s privilegiji (pravicami) z uporabo enega samega sistema ali aplikacije je široko sprejeta dobra praksa. Aplikacije, kot so Oracle DBMS, PostgreSQL 8.1, SAP R/3, ISIS Papyrus, FusionForge, Wikipedia, Microsoft Lync, Microsoft Active Directory, Microsoft SQL Server in operacijski sistemi Linux, Solaris, grsecurity (Linux), TrustedBSD (FreeBSD) in še veliko drugih, uspešno uporabljajo tak način upravljanja s pravicami. Poročilo pripravljeno za NIST[35] v letu 2010 s strani organizacije Research

Triangle Institute prikazuje analizo, ki kaže velike prednosti uporabe RBAC v velikih podjetjih. Z uporabo RBAC pridobijo zmanjšano delovno odsotnost, bolj učinkovito upravljanje in večji nadzor nad politiko upravljanja. V organizaciji s heterogeno infrastrukturo in zahtevami, ki pokrivajo več sto sistemov in aplikacij, je uporaba RBAC brez hirearhične zasnove vlog zelo kompleksna za upravljanje[14]. Novi sistemi razširijo star NIST RBAC model[40], da obidejo omejitve RBAC v velikih sistemih. Model NIST je bil prilagojen kot standard s strani organizacije INCITS kot standard ANSI/INCITS 359-2004. Diskusija o možnih izvedbah modela NIST je tudi objavljena[12].

2.2.4.3 Primerjava med RBAC in ACL

Prva alternativa modelu RBAC je model ACL. Minimalen RBAC model, imenovan RBACm, je primerljiv z mehanizmom ACL, kjer so v seznamu pravic dovoljene le grupe (imenovan ACLg). Raziskava[5] je pokazala, da sta model RBACm in model ACLg enakovredna. V modernih implementacijah lahko ACL upravlja z grupami in odvisnostmi na hirearhičen način (npr. ACL pri CakePHP). Torej moderna implementacija ACL se lahko primerja z moderno implementacijo RBAC, bolj kot s staro različico, ki je zasnovana na datotečnem sistemu.

2.3 Avtentikacija

V tem poglavju bomo govorili o avtentikaciji, o tehničnih podrobnostih in še posebej o vidikih varnosti, ki so zelo pomembni.

Pri uporabi spletnih aplikacij je pomembna uporabnikova izkušnja. Še posebej, če spletna aplikacija vsebuje sistem prijave in nerodno bi bilo, v kolikor bi vedno znova od uporabnika zahtevali, da se v sistem prijavi. Večina spletnih uporabnikov danes pozna ali razume samo postopek prijave in odjave. Prijavo uporabnik tipično uporabi, ko nekaj želi narediti in odjavo, ko s temi aktivnostmi želi končati. Vemo, da avtentikacija danes predstavlja več kot samo to. Danes spletne aplikacije pogosto vključujejo avtentikacijo tretjega – zunanjega ponudnika, kot so npr. OAuth, Facebook Connect, Open ID.

Ko je Facebook objavil novico o tehnologiji Facebook Connect, je ta stran postala prvi zadetek na iskalnikih, ko so ljudje iskali Facebook login. Veliko ljudi uporablja iskalnik za navigacijo na spletno stran. Nekateri uporabniki so bili prepričani, da gre za prenovljeno spletno stran Facebook in so izvedli avtentikacijo, čeprav so storili nekaj popolnoma drugega. Uporabniki so bili že prijavljeni v spletno omrežje Facebook, z izvedeno avtentikacijo pa so povezali spletno stran s svojim Facebook računom. Tako ima spletna aplikacija še danes dostop do podatkov o teh računih.

Raziskava kaže[38], da 38% uporabnikov sistema Twitter uporablja več kot en uporabniški račun. Podobno je Google omogočil prijavo z več kot enim uporabniškim računom. Slabost navadne avtentikacije z uporabniškim računom in geslom je prenos teh podatkov po omrežju. Na tej osnovi je bil zgrajen OAuth, Twitter se je prilagodil, Facebook je uvedel svojo različico prijave v sistem, kasneje pa se je prav tako prilagodil na način, da podpira prijavo z OAuth 2.0 različico.

Več ljudi, več prometa, več povezav. Uporabniške spletne strani, ki uporabljajo Facebook Connect, beležijo 30 – 200% rast[38] števila ljudi, ki so se registrirali. Uporabniku ni treba imeti velikega števila gesel. V primeru pozabljenega gesla je uporabnik preusmerjen k ponudniku storitve avtentikacije. Take rešitve pospešijo razvoj spletnih aplikacij. Prihaja celo do zmede. Zamislite si, da se v spletno stran Techcrunch prijavite z uporabniško identiteto Yahoo, ki ste jo kreirali z uporabniško identiteto Facebook. Lahko bi rekli, da to niti nima več nobenega smisla.

Bolj ko je organizacija odvisna od zunanjih ponudnikov, bolj kritična je infrastruktura organizacije. Dosegljivost storitev organizacije je naenkrat odvisna tudi od dosegljivosti storitev ponudnika. Facebook ima na primer predvidoma večjo dosegljivost storitev kot naša spletna stran. Vseeno se moramo zavedati, da se nedosegljivost njihovega sistema prišteva k nedosegljivosti našega sistema. In tu ne moremo nič narediti. Tu pride tudi do pomanjkanja kontrole uporabnikov. Razvijalec spletne strani lahko zahteva pravice, ki jih uporabniki ne bi želeli. Odlično bi bilo, če bi uporabniku lahko ponudili, da sam izbere pravice, ki jih želi deliti. In tu igrajo pomembno vlogo moderni sistemi za avtentikacijo.

Ponuditi uporabniku samo en način avtentikacije je najbrž slabost. Pojavlja se vprašanje, ali uporabnik resnično želi izpostaviti svoje uporabniške podatke, da obišče določeno spletno stran. Vedno moramo imeti tudi rezervni scenarij. Na kakšen način bomo obvestili uporabnike o spremembi, v kolikor zunanji uporabnik ne nudi več storitev na enak način. Rezervni scenarij vključuje pridobitev elektronskega naslova uporabnikov, da v primeru spremembe lahko stopi v stik z uporabnikom. Več zunanjih storitev imamo, težje se prilagodimo in več scenarijev moramo pokriti. Nekatera družabna omrežja od uporabnikov zahtevajo samo določene pravice, ki imajo smisel. Kaj pa spletne aplikacije, ki želijo dostop do podatkov celotne družine ali npr. video posnetkov? S takimi in podobnimi težavami se danes srečujemo pri uporabi zunanjega ponudnika za upravljanje z identitetami.

Smo v procesu, ko se modeli upravljanja z identitetami spreminjajo in vsi, ki uporabljamo družabna omrežja smo tisti, ki bomo vplivali na njihov razvoj. V nadaljevanju si bomo ogledali različne tipe avtentikacije. Ena izmed družin protokolov za avtentikacijo uporablja mehanizem vprašanje – odgovor (challenge-response), kjer ena stran predstavi vprašanje

(»challenge«), druga stran pa poda pravilen odgovor (»response«), da se lahko avtenticira. Med najbolj enostavne protokole za avtentikacijo tipa »challenge-response« štejemo avtentikacijo z geslom (angleško password authentication), kjer ena stran sprašuje za geslo in pravilen odgovor je veljavno geslo. Seveda nekdo, ki prisluškuje omrežju lahko uporabi isto informacijo za izvedbo avtentikacije tretje osebe. Ena izmed rešitev je uporaba več gesel, kjer je vsako geslo označeno tudi z identifikatorjem. Strežnik, ki preverja avtentikacijo, lahko zahteva katerokoli geslo in odjemalec mora podati pravilen odgovor. Ponekod uporabljajo številko oziroma kodo, s katero šifrirajo del informacije. Za dodatno varnost lahko kodi določimo čas veljavnosti, tipično 24 ur. Gre za mehanizme, ki preprečujejo robotiziranim odjemalcem, da bi samodejno izpolnjevali obrazce in zagotovijo, da gre na drugi strani za pravo identiteto.

Protokoli brez šifriranja so bili lahko zadovoljivi pred uporabo v spletu, danes pa imamo rešitve, ki uporabljajo kriptografijo in dvo smerno avtentikacijo. Za dodatno varnost protokol zahteva, da obe strani dobita potrdilo, da druga stran pozna skupno skrivnost oziroma geslo. To se izvede na način, da skrivnost nikoli ni predmet prenosa po komunikacijskih kanalih, kjer se lahko izvaja prisluškovanje. Eden izmed načinov implementacije je z uporabo gesla za šifriranje podatkov in prenos naključno generirane informacij kot izziv (»challenge«), medtem ko mora druga stran vrniti odgovor na podobno šifriran način, s čimer dokaže, da zna to informacijo odšifrirati. Dodatno je šifrirana vrednost transformirana z vnaprej znano funkcijo, s katero je bil podan tudi prvotni zahtevek. S tem dosežemo enoličnost generirane informacije in preprečujemo napade med komunikacijo in napade s posnemanjem prometa na komunikacijskih kanalih, saj je geslo vsakič drugačno. Uporaba časovnih funkcij tu ni priporočljiva, saj predstavljajo šibko zaščito, če so strežniki oddaljeni, saj je časovna napaka lahko velika. Medsebojna avtentikacija izvaja rokovanje v obeh straneh in s tem zagotavlja odjemalcu, da strežnik pozna skrivnost in obratno. To preprečuje zlorabe, ko napadalec želi poosebljati pravi strežnik.

Avtentikacija implementirana na način vprašanje – odgovor (challenge response) lahko reši težavo pri izmenjavi ključa za sejo za izvedbo šifriranja. S kombinacijo ključa, ki ga generira funkcija, in skrivnostjo postane ključ nepredvidljiv za sejo. To je še posebej resnično v primerih napada med komunikacijo, saj napadalec ne more generirati pravilnega ključa, če ne pozna skrivnosti in tako ne more dešifrirati toka podatkov.

2.3.1 Avtentikacija HTTP

Protokol HTTP vključuje osnovno različico protokola za avtentikacijo (basic access) in napredno različico protokola (digest access), ki omogočata dostop do spletne strani, če je

uporabnik podal pravilno uporabniško ime in geslo. V kolikor strežnik zahteva vnos uporabniškega imena in gesla za dostop do spletne strani, brskalnik to zahteva od uporabnika in ko uporabnik podatke vnese, brskalnik informacije shrani in jih pošlje v vsaki naslednji zahtevi na stran. S to informacijo lahko sledimo uporabniku.

2.3.2 Osnovna različica protokola za avtentikacijo (Basic access)

V kontekstu transakcije HTTP je osnovna različica avtentikacije (v angleškem jeziku Basic access authentication ali s kratico BA) metoda, ki preko uporabnikovega agenta HTTP poda uporabniško ime in geslo pri izvedbi spletne poizvedbe. Gre za najbolj enostavno različico za zagotavljanje kontrole dostopa do virov na spletni strani, saj ne uporablja piškotkov, identifikacije seje in strani za prijavo, pač pa uporablja statično metodo s podajanjem informacij v glavi poizvedbe HTTP. To pomeni, da protokol ne uporablja rokovanja za izmenjavo informacij na varen način. Protokol imenujemo tudi šibek protokol, saj se informacije prenašajo na način, ki so v primeru prisluškovanja povezavi razvidne kot prosti tekst oziroma niso zaščitene.

2.3.2.1 Izvedba na strani strežnika

Ko strežnik želi, da se uporabniški agent avtenticira, pošlje zahtevo za avtentikacijo uporabniku. Zahteva se pošlje z odgovorom »HTTP 401 Not Authorized« in v glavi poizvedbe HTTP vsebuje tekst »WWW-Authenticate« ter je največkrat v obliki:

```
WWW-Authenticate: Basic realm="WallyWorld"
```

2.3.2.2 Izvedba na strani odjemalca

Ko uporabniški agent želi posredovati uporabniško ime in geslo, lahko uporabi glavo za avtorizacijo. Ta je sestavljena po naslednjih navodilih:

- uporabniško ime in geslo združimo v obliko "uporabnik:geslo",
- združeni niz v zgornjem primeru transformiramo z uporabo RFC 2045[26] MIME variacije Base64 kodiranja,
- pred zgoraj sestavljeni niz dodamo še metodo za avtorizacijo in presledek, npr. »Basic «.

Primer tako sestavljenega niza je:

```
Authorization: Basic Agdk2chkbFpvD1duIHNlc2EtDq==
```

2.3.3 Avtentikacija na osnovi HTTP in HTML obrazcev

Avtentikacija HTTP v kombinaciji z obrazcem HTML je tehnika, kjer spletna stran uporabi obrazec za zbiranje podatkov potrebnih za avtentikacijo. Koraki za izvedbo avtentikacije zajemajo:

- neavtoriziran uporabniški agent izvede poizvedbo na spletno stran preko protokola HTTP,
- spletna stran vrne vsebino uporabniškemu agentu, ki vsebuje obrazec z vnosnim poljem za uporabniško ime in geslo,
- uporabnik vnese uporabniško ime in geslo in pritisne gumb za potrditev,
- uporabniški agent pošlje vsebino obrazca, ki vsebuje tudi podatke uporabniškega imena in gesla strežniku,
- strežniški del spletne strani izvede preverjanje pravilnosti in določi uporabnika kot avtenticiranega.

Enostaven način avtentikacije danes, skupaj z uporabo šifriranja pri uporabi protokola HTTPS, rešuje veliko nevarnosti pri uporabi, kot to rešuje tako imenovana avtentikacija »Digest access« opisana v nadaljevanju.

2.3.4 Avtentikacija »Digest access«

Način avtentikacije »Digest access« je ena izmed dogovorjenih metod avtentikacije v spletnem brskalniku z uporabo rokovanja z občutljivimi podatki, kot so uporabniško ime in geslo. Rokovanje se uporabi, da se potrdi identiteto uporabnika, preden se posredujejo občutljive informacije. Funkcija hash transformira vrednost gesla, preden posreduje podatek v omrežje, kar predstavlja bolj varen način kot način avtentikacije »basic access«, ki posreduje navaden tekst. Način »digest access« uporablja HTTP protokol za izmenjavo podatkov, vendar za izvedbo avtentikacije uporabi funkcijo za šifriranje MD5 in funkcijo hash.

Specifikacija za avtentikacijo "Digest access" je prvotno podana v RFC 2069 [29]. Odgovor avtentikacije je podan v obliki, kot prikazuje Tabela 1.

Kasneje je bil standard nadomeščen z novim RFC 2617[28], ki vpeljuje številne nove možnosti in izboljšave v kvaliteti zaščite (direktiva »qop« izvira iz quality of protection) zagotavljanja varnosti. Ena izmed prednosti je, da je naključni ključ generiran na strani odjemalca, kjer se izvaja tudi povečevanje vrednosti podane v naključnem ključu (v tabeli

imenovan nonce). Tabela 1 prikazuje, kako se sestavi odgovor glede na uporabljeni standard in v tabeli je razvidno, da v primeru, da parameter kvaliteta zaščite ni podan (»qop«), je uporabljen novejši standard. Odgovor je torej odvisen od direktive metoda in kvalitete zaščite.

RFC standard	**metoda**	**Qop**	**HA1**	**HA2**	**Odgovor**
2069 [29]	prazen ali »MD5«	/	MD5(username:realm: password)	MD5(method:digest URI)	MD5(HA1:nonce:HA2)
2069	»MD5-sess«	/	MD5(MD5(username: realm: password):nonce: cnonce)	MD5(method: digestURI)	MD5(HA1:nonce:HA2)
2617 [28]	prazen ali »MD5«	prazen ali »auth«	MD5(username:realm :password)	MD5(method:digest URI)	MD5(HA1:nonce:nonce Count:clientNonce:qop: HA2)
2617	»MD5-sess«	prazen ali »auth«	MD5(MD5(username:realm: password):nonce: cnonce)	MD5(method:digest URI)	MD5(HA1:nonce:nonce Count:clientNonce:qop: HA2)
2617	prazen ali »MD5«	»auth-int«	MD5(username:realm :password)	MD5(method: digestURI:MD5(entityBody))	MD5(HA1:nonce:nonce Count:clientNonce:qop: HA2)
2617	»MD5-sess«	»auth-int«	MD5(MD5(username:realm: password):nonce: cnonce)	MD5(method: digestURI:MD5(entityBody))	MD5(HA1:nonce:nonce Count:clientNonce:qop: HA2)

Tabela 1: Prikaz sestave odgovora v primeru »digest access« glede na vhodne parametre

2.3.4.1 Prednosti in slabosti

Imamo dve glavni prednosti:

- avtentikacija »HTTP digest« je bolj varna kot tradicionalna avtentikacijska shema oziroma bolj močna kot npr. CRAM-MD5 [28],
- geslo ni posredovano v berljivi obliki, ampak je transformirano preko funkcije MD5, konkretno v obliki MD5(username:realm:password). To omogoči določenim implementacijam, da namesto gesla v konfiguracijskih datotekah uporabijo izračunano vrednost.

Naštejmo še nekaj slabosti:

- veliko parametrov v novejši specifikaciji RFC 2617[28] ni obveznih; če parameter »kvaliteta zaščite« ni podan s strani strežnika, bo odjemalec uporabil za izvedbo avtentikacije obliko, ki je manj varna,
- avtentikacija je ranljiva pri napadu komunikacije tipa man-in-the-middle (MitM),
- avtentikacija nima mehanizma za preverjanje identitete strežnika.

Avtentikacija »digest access« je nastala z namenom, da nadomesti standardno nezaščiteno avtentikacijo. Njen namen ni nadomestiti močnih protokolov za avtentikacijo, kot so javni ključ ali avtentikacija Kerberos, kateri si bomo pobližje ogledali v nadaljevanju.

2.3.4.2 Shranjevanje gesel

Da se izognemo shranjevanju gesel, določeni operacijski sistemu shranjujejo transformirano geslo z zgoščevalno (hash) funkcijo. Med avtentikacijo se vnešeno geslo pretvori preko iste funkcije in primerjava se izvede šele potem. Na tak način je napadalcu težje uganiti geslo, saj ni shranjeno v prvotno podani obliki. To pa predstavlja problem pri tako imenovanih »challenge-response« algoritmih, ki zahtevajo, da skupno skrivnost hranita strežnik in odjemalec. Ker geslo v prvotni obliki ni nikjer shranjeno, morajo taki algoritmi pogosto uporabiti kar transformirano geslo, namesto prvotno podanega gesla. V takem primeru napadalec lahko uporabi že transformirano vrednost, ki predstavlja enako nevarnost kot samo geslo.

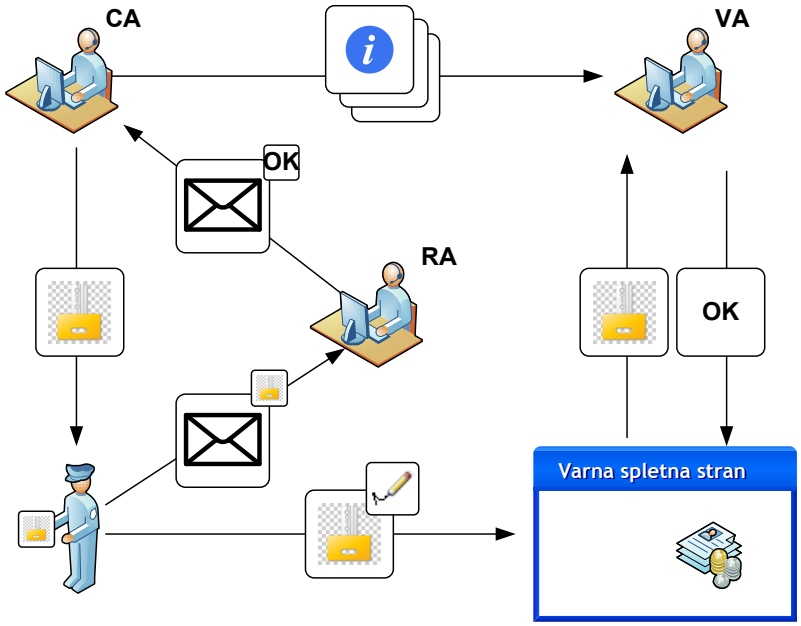
2.3.5 Protokoli za avtentikacijo z uporabo šifriranja

Protokoli za avtentikacijo s podporo šifriranja, pogosto imenovani tudi močni protokoli za avtentikacijo v spletnih aplikacijah, so:

- avtentikacija PKI (Public key authentication) ali avtentikacija z javnim ključem je ponavadi implementirana s HTTPS / SSL certifikatom na strani odjemalca,
- avtentikacija Kerberos ali SPNEGO, najprej uporabljena v spletnem strežniku IIS podjetja Microsoft za vgrajeno avtentikacijo Windows (Integrated Windows Authentication ali IWA),
- protokol za varno oddaljeno geslo (Secure Remote Password protocol), ponavadi znotraj nivoja HTTPS / TLS.

2.3.6 Asimetrični protokol PKI

Infrastruktura javnih ključev (v angleškem jeziku Public Key Infrastructure ali s kratico PKI) je niz strojne opreme, programske opreme, ljudi, politik in postopkov, ki so potrebni za ustvarjanje, upravljanje, distribucijo, uporabo, shranjevanje in preklic digitalnih potrdil. V kriptografiji je infrastruktura javnih ključev dogovor, ki veže javne ključe z ustreznimi uporabniškimi identitetami s pomočjo overitelja potrdil (v angleškem jeziku »Certificate Authority« ali s kratico v nadaljevanju CA). Identiteta uporabnika mora biti enolična v okviru domene overitelja. Tretja oseba, ki sme izvajati overjanje (v angleškem jeziku »Validation Authority« ali s kratico VA), lahko podaja to informacijo v imenu overitelja digitalnih potrdil. Povezava je vzpostavljena skozi proces registracije, ki je odvisen od stopnje zaupanja pri vzpostavljanju povezave, in se izvede s progamsko opremo pri CA ali pod človeškim nadzorom. Vloga PKI, ki zagotavlja to povezavo, se imenuje organ za registracijo (Registration authority ali s kratico RA) in zagotavlja, da je javni ključ vezan na posameznika na način, da ne pride do zavračanja. Imamo tri pristope, da pridobimo zapuanje: overitelj digitalnih potrdil (CA), spletno zaupanje (web of trust ali WoT) in poenostavljen PKI - simple PKI oziroma SPKI.



Slika 3: PKI infrastruktura

2.3.6.1 Overitelj potrdil (CA)

Primarna vloga overitelja potrdil (certificate authority ali CA) je, da digitalno podpiše in objavi javni ključ, ki je povezan z danim uporabnikom. To se izvede s privatnim ključem overitelja, tako da je zaupanje do ključa odvisno od zaupanja veljavnosti ključa overitelja. Ko je overitelj tretja stranka, ki je ločena od uporabnika in sistema, ga imenujemo organ za registracijo (registration authority ali RA), ki je lahko ločen od overitelja ali pa tudi ne. Povezava med ključem in uporabnikom, ki je vzpostavljena, je odvisna od stopnje zaupanja. Pojem zaupljiva tretja stranka (trusted third party ali TTP) je prav tako lahko uporabljen za overitelja potrdil (CA). Infrastruktura javnih ključev PKI je zato pogosto uporabljena kot sinonim za implementacijo CA.

2.3.6.2 Začasno digitalno potrdilo in enotna prijava

Ta prijem vključuje strežnik, ki ima vlogo overitelja potrdil v okviru sistema za enotno prijavo. Strežnik za enotno prijavo (single sign-on server) izda digitalno potrdilo v sistem odjemalca, vendar ga nikoli ne shrani. Uporabnik lahko izvaja program ali drugo opravilo z začasnim digitalnim potdilom.

2.3.6.3 Omrežje zaupanja (Web of Trust ali WoT)

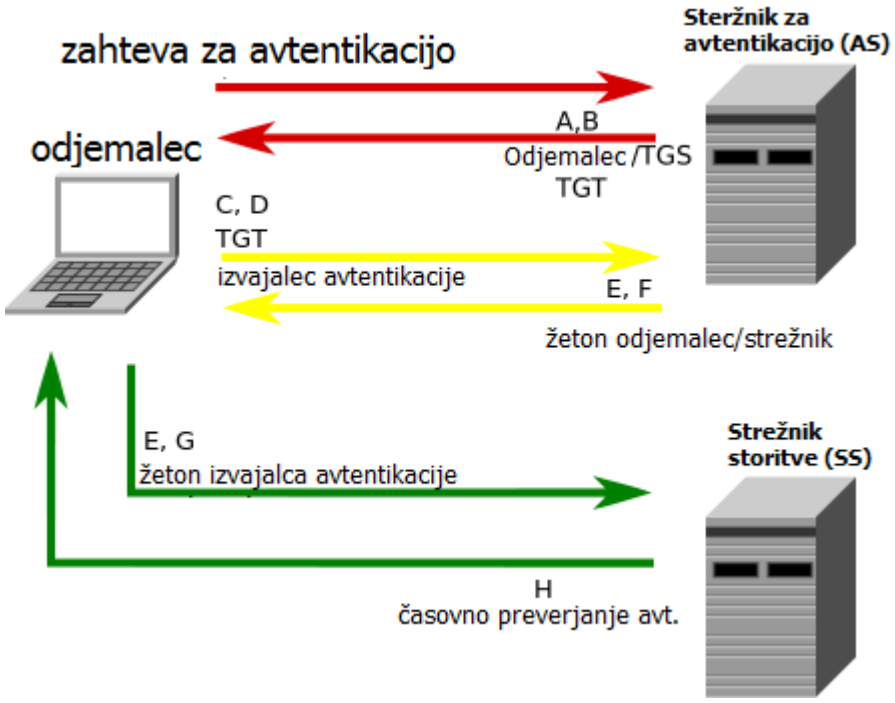
Alternativni pristop k težavi javne avtentikacije z javnim ključem je shema imenovana »Web of Trust« ali s kratico WoT. Pristop uporablja lastnoročno podpisano digitalno potrdilo in tretjo stranko, ki potrdi digitalna potrdila. Primera take uporabe sta PGP (Pretty Good Privacy) and GnuPG (implementacija OpenPGP, ki je standardizirana po specifikaciji PGP). Prednost takega pristopa je, da zaupamo vsem strankam znotraj domene, kot je npr. overitelj znotraj organizacije, ki garantira, da je potrdilo vredno zaupanja. Če je omrežje zaupljivo, potem zaupati potrdilu pomeni, zaupati vsem potrdilom v omrežju zaupanja. To pravzaprav pripelje do anomalije. S časom se ključi od drugih strank, katere določimo kot zaupljive, kopičijo. Ostale stranke bodo ustvarile svoj krog zaupanja. S časom se bodo ključi nabirali in uporabljali v omrežju vključno z njihovimi digitalnimi podpisi s pričakovanjem, da bo vsakdo, ki bo ključ in potrdilo prejel, zapual vsaj enemu ključu. To pripelje do decentralizacije in nezaupanja javnim ključem.

2.3.6.4 Poenostavljen PKI (Simple public key infrastructure – SPKI)

Gre za alternativo, ki nima neposredne povezave s protokolom PKI, ampak se je razvila neodvisno od nje z namenom, da poenostavi kompleksnost X.509 in WoT. Protokol SPKI ne povezuje uporabnikov z osebami, saj je ključ tisto, v kar zaupa. Protokol SPKI ne vsebuje pojma zaupanja, saj je tisti, ki preverja tudi izdajatelj.

2.3.7 Simetrični protokol KERBEROS

V protokolu Kerberos je poizvedba šifrirana s številko N, odgovor pa s številko N+1 in s tem zagotavljamo, da je druga stran pravilno izvedla dešifriranje. V določenih različicah se uporablja zgoščevalna (hash) funkcija nad geslom in naključno generirano število za izgradnjo odgovora. Na tak način geslo ni razvidno v primeru prisluškovanja povezavi, vendar pa lahko ponudi dovolj priložnosti za določene tipe napadov, kot so napad na silo (brute force attack) ali napad z uporabo slovarja (dictionary attack). Informacija, ki je šifrirana z naključno vrednostjo ob vsaki izmenjavi podatkov, varuje pred napadi s ponavljanjem vsebine (replay attack), kjer napadalec posname vsebino in jo kasneje ponovno uporabi za namen zlorabe avtentikacije. Protokol omogoča varno komunikacijo na nezaščitenem kanalu. Pri izvedbi avtentikacije tako odjemalec in strežnik preverita identiteto drugega. Slika 4 prikazuje, na kakšen način poteka pogajanje pri izvedbi avtentikacije. Avtentikacija z uporabo protokola Kerberos sloni na simetričnem ključu za kriptografijo in zahteva tretjo stranko. Dodatno lahko v določenih korakih pri izvedbi avtentikacije uporabimo javne šifrirane ključe. Sporočila pri uporabi protokola Kerberos so zaščitena.



Slika 4: Pogajanje pri Kerberos načinu avtentikacije

Poglejmo si potek prijave s strani odjemalca:

- uporabnik vnese uporabniško ime in geslo,
- odjemalec transformira geslo v simetrični ključ, ki je osnovan na času ali enosmerni zgoščevalni funkciji.

Sledi avtentikacija odjemalca:

- Odjemalec pošlje oznako uporabnika strežniku za avtentikacijo za uporabo storitev v imenu uporabnika. Strežniku se ne pošlje niti skrivni ključ niti geslo. Strežnik za avtentikacijo generira skrivni ključ z uporabo zgoščevalne funkcije nad geslom uporabnika, ki ga pridobi iz baze, kot je npr. Active Directory.
- Strežnik za avtentikacijo preveri, ali je uporabnik v podatkovni bazi in v primeru zadetka pošlje odjemalcu dve sporočili:
  - sporočilo A: ključ seje odjemalca šifriran s skrivnostjo odjemalca,
  - sporočilo B: žeton za dostop, ki vsebuje oznako uporabnika, uporabnikov naslov v omrežju, veljavnost žetona, in ključ seje odjemalca.
- Uporabnik prejme sporočili A in B ter poskuša sporočilo A dešifrirati s skrivnim ključem generiranim iz gesla, ki ga je vnesel uporabnik. Če je geslo uporabnika različno od tistega, ki je zapisano v podatkovno bazo, potem se ključ ne bo ujemal in odjemalec ne bo mogel dešifrirati sporočila A. Sicer uporabnik z dešifriranjem sporočila A pridobi ključ seje. Ta se od tega trenutka uporablja za izvedbo komunikacije.

Od tega trenutka naprej ima uporabnik vse predpogoje za izvedbo avtorizacije:

- Ko odjemalec zahteva storitev, pošlje strežniku sporočilo C in D:
  - sporočilo C: vsebuje žeton za dostop, pridobljen v sporočilu B, in oznako storitve,
  - sporočilo D: sestavljeno iz oznake uporabnika in časovne znamke, šifrirano s ključem seje.

- Ko strežnik prejme sporočili C in D, pridobi sporočilo B iz sporočlila C. Sporočilo B dešifrira z žetonom za dostop. Na ta način strežnik pridobi ključ seje odjemalca. S tem ključem strežnik dešifrira sporočilo D in pošlje odjemalcu dve sporočili:
    - sporočilo E: žeton »odjemalec – strežnik«, ki vsebuje oznako uporabnika, naslov odjemalca v omrežju, veljavnost žetona in ključ seje, vse skupaj šifrira s skrivnim ključem storitve,
    - sporočilo F: ključ seje »odjemalec – strežnik«, šifriran z žetonom dostopa.

Sledi še potek same poizvedbe:

- Po prejetju sporočil E in F s strežnika ima odjemalec dovolj informacij, da se lahko uspešno avtenticira v strežnik storitve.Odjemalec se poveže s strežnikom storitve in pošlje:
    - sporočilo E,
    - sporočilo G: vključuje oznako uporabnika ter časovno znamko šifrirano s ključem seje »odjemalec – strežnik«.
- Strežnik storitve dešifrira žeton z uporabo lastnega skrivnega ključa in tako pridobi ključ seje »odjemalec – strežnik«. S tem ključem dešifrira sporočilo G in pošlje naslednje sporočilo, s katerim potrdi identiteto:
    - sporočilo H: časovna znamka iz sporočila G, šifrirano s ključem seje »odjemalec – strežnik«.
- Odjemalec dešifrira potrditev z uporabo ključa seje »odjemalec – strežnik« in preveri, če se je časovna znamka pravilno spremenila. Če so izpolnjeni vsi pogoji, potem odjemalec lahko zaupa strežniku in lahko prične s poizvedbami za uporabo storitev.

2.3.8 Protokol Secure Remote Password

Protokol »Secure Remote Password« ali s kratico SRP je razširjena oblika avtentikacije z geslom, ki uporablja protokol PAKE (password-authenticated key agreement). Pri uporabi protokola PAKE smo zaščiteni na napade med omrežjem ali prisluškovanje, ker napadalec ne more pridobiti gesla na »nasilen način (angleško brute force). Uporaba protokola nudi povečano varnost v primeru uporabe šibkih gesel. Strežnik v primeru uporabe protokola ne

shranjuje gesla v prvotni obliki, kar dodatno otežuje napadalcu delo, saj mora najprej pridobiti geslo na nasilen način. Protokol ima naslednje dobre lastnosti:

- omogoča uporabniku, da se sam prijavi na strežnik,
- imun je na napade z uporabo slovarja,
- ne potrebuje tretje stranke.

Protokol uporablja velik privatni ključ, ki je deljen med dve stranki. Na strani odjemalca imamo uporabnikovo geslo, na strani strežnika pa kriptografski algoritem za preverjanje, izpeljan iz gesla. Deljen ključ je sestavljen iz dveh naključnih števil. Eno generira odjemalec, drugo strežnik in sta enolični glede na poizkus prijave. V primerih, ko je zahtevano šifriranje na nivoju komunikacije in avtentikacije, protokol nudi večjo zaščito kot alternativni protokol SSH. Prav tako ni odvisen od tretjih strank, kot je npr. Kerberos. Različica 3 protokola je definirana v specifikaciji RFC 2945[33]. Protokol se uporablja tudi v kombinaciji s protokolom SSL/TLS in je povezljiv z nekaterimi standardi, kot je SAML[34] in standardiziran v IEEE P1363[16] in ISO/IEC 11770-4.

2.3.9 Avtentikacija na spletnih straneh

2.3.9.1 Enotna prijava (Single sign-on ali SSO)

Drugi načini avtentikacije, ki jih ne smemo mešati z enotno prijavo, vključujejo OAuth, OpenID Connect in Facebook Connect, ki zahtevajo od uporabnika, da se prijavi vsakič, ko dostopa do druge spletne strani oziroma aplikacije. Enotna prijava je lastnost nadzora dostopa večih med seboj odvisnih sistemov. Z enotno prijavo se uporabnik prijavi samo enkrat in s tem pridobi dostop do vseh sistemov, brez da bi bilo treba izvesti ponovno prijavo v sistem. Ponavadi se za ta namen uporabi protokol LDAP in shranjevanje podatkovnih baz LDAP na strežnikih. Enostavno različico enotne prijave lahko dosežemo z uporabo piškotkov, ampak samo v primeru, da sta spletni strani v isti domeni. Nekaj implementacije enotne prijave[52]:

- Active Directory Federation Services,
- Facebook connect,
- Forefront Identity Manager,
- FreeIPA (Red Hat),
- IBM Tivoli Access Manager,

- Protokol Kerberos,
- Microsoft account,
- SAML.

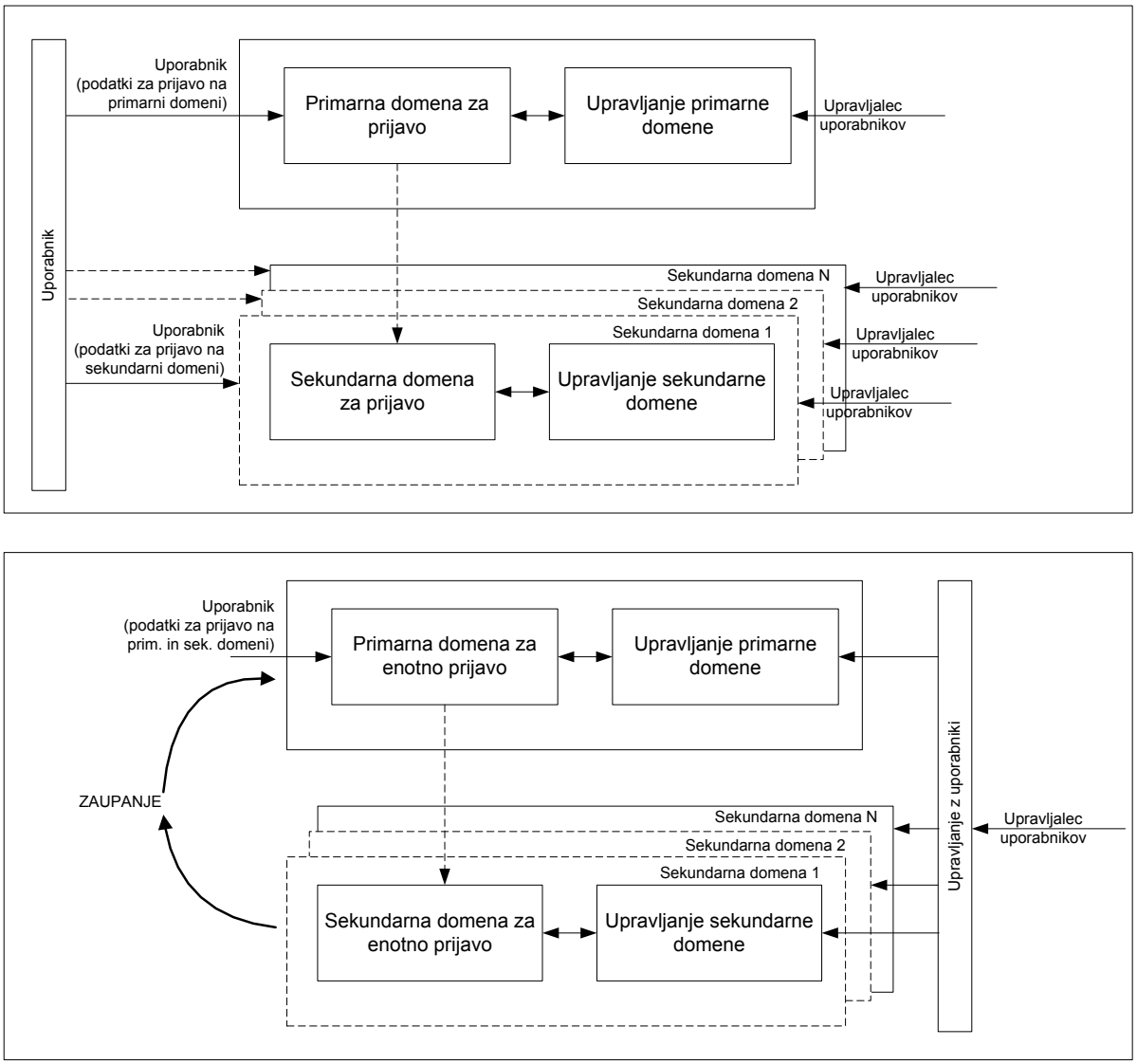
Nekateri ponudniki podpirajo tudi uporabo OpenId, OAuth ali SAML, kar povečuje združljivost z odprtokodnimi implementacijami avtentikacije in z avtentikacijami, ki se uporabljajo v družabnih omrežjih.

Prednosti enotne prijave

Prednosti enotne prijave[1] so zajete na več področjih:

- uporabniška izkušnja (najbolj pomemba prednost je, da uporabnik lahko prehaja med aplikacijami na varen način, brez da bi moral vsakič znova vpisati svoje uporabniško ime in geslo; sistem enotne prijave združi posamezne storitve v portal in odstrani omejitve storitev – preklop med različnimi aplikacijami je za uporabnika transparenten),
- varnost (zaupni podatki uporabnikov so shranjeni na strežniku SSO in ne na strežniku, ki nudi storitev),
- zmanjšanje porabe virov (zaradi centraliziranega upravljanja preko spletne aplikacije),
- povezljivost (aplikacije in spletni razvijalci imajo dostop do vmesnikov za vgradnjo celotne avtentikacije in avtorizacije v spletne aplikacije ter storitve na varen način).

Slika 5 prikazuje primerjavo med prijavo v več sistemov brez enotne prijave in z uporabo enotne prijave.



Slika 5: Primerjava prijave v več sistemov z uporabo enotne prijave (spodaj) in brez (zgoraj)

2.3.10 Avtentikacija v družabnih omrežjih

Uporabniki družabnih omrežij ponavadi niso programerji in ne poznajo varnostnih mehanizmov. Pomembno je tudi, da gre pri družabnih omrežjih za maso ljudi, ki nimajo veliko tehnološkega znanja. Vmesnik za avtentikacijo mora biti zato čim bolj enostaven in prijazen. Vseeno pa mora vmesnik zagotavljati varnost, saj se zaradi ogromnega števila uporabnikov tudi povečujejo možnosti zlorabe. Najprej si oglejmo tehnologijo, ki jo danes srečamo pri vseh izvedbah avtentikacije v družabnih omrežjih in jo lahko štejemo tudi kot dopolnitev k obstoječim protokolom.

2.3.11 Dopolnitev k protokolom – piškotki

V celotni zgodbi avtentikacije zagotovo ne moremo iti mimo piškotkov (cookies), saj igrajo pomembno vlogo pri zagotavljanju boljše storitve uporabniku in lahko rečemo, da gre za dopolnitev k protokolom. Uporabljajo se v večini izvedb avtentikacije. V nadaljevanju si poglejmo vrste piškotkov.

2.3.11.1 Piškotek za sejo (session cookie)

Piškotek za sejo, znan tudi kot piškotek, ki se nahaja v spominu računalnika, je v spominu le določen čas in to za čas obiska določene spletne strani. Ko poteče veljavnost obstoječega ali pa ko interval preverjanja veljavnosti piškotka ni določen, je piškotek kreiran. Brskalnik tipično pobriše tak piškotek, ko uporabnik zapre brskalnik.

2.3.11.2 Obstojen piškotek (persistent cookie)

Obstojen piškotek zdrži dlje kot je čas veljavnosti določene seje. Če ima določen čas veljavnosti eno leto, bo piškotek pošiljal prvotno nastavljeno vrednost celo leto, in to vsakič, ko bo uporabnik obiskal spletno stran. To bi prišlo v poštev v primeru, ko bi radi zapisali ključne informacije o tem, kako je določen uporabnik prvotno prišel na to spletno stran. To je razlog, da to vrsto piškotkov imenujemo tudi sledljivi piškotki (tracking cookies).

2.3.11.3 Varen piškotek (secure cookie)

Varen piškotek ima lastnost za varnost nastavljeno na vrednost »varen« in se uporablja samo v primeru uporabe protokola HTTPS, kar zagotavlja, da je piškotek vedno šifriran, ko se prenaša med odjemalcem in strežnikom. To zagotavlja manjšo verjetnost, da bi s prisluškovanjem povezave lahko pridobili oz. ukradli vrednost piškotka. Ta bi lahko vseboval tudi podatke o identiteti.

2.3.11.4 HttpOnly piškotek (HttpOnly cookie)

Gre za lastnost piškotka, ki ga podpira večina modernih brskalnikov. V brskalnikih, ki podpirajo tak piškotek, se piškotek uporablja le pri dostopu preko HTTP ali HTTPS poizvedbe. Dostop do piškotkov z uporabo programskega jezika JavaScript ni mogoč. Ta omejitev sicer deloma odstrani možnost za krajo priškotka, v celoti pa te možnosti ne odpravlja. Ta vrsta piškotka se uporablja v kontekstu seje uporabnika.

2.3.11.5 Piškotek tretje osebe (third-party cookie)

Prvo osebni piškotek je tisti, ki pripada samo določeni domeni, ki se pojavi v naslovni vrstici brskalnika. Piškotek tretje osebe je piškotek, ki velja za domeno, ki ni navedena v naslovni vrstici brskalnika. Spletne strani lahko gostijo vsebino, ki jo ponuja druga domena (tak primer

so oglasi), vendar tak pristop odpira možnosti, da uporabniku sledimo oz. sledimo zgodovini obiskanih spletnih strani. Večina današnjih brskalnikov omogoča blokiranje takih piškotkov.

2.3.11.6 Super piškotek (supercookie)

Super piškotek se navezuje na najvišji nivo domene, kot je npr. ».si«. Pomembno je, da se tak piškotek blokira s strani brskalnika, saj odpira varnostne luknje. Če uporaba takega piškotka ni blokirana, lahko napadalec nastavi tak piškotek in s tem poruši zakonitost zahtevkov do drugih spletnih strani, ki uporabljajo isti naziv za najvišji nivo domene. Na primer, super piškotek na domeni ».si« bi lahko kvarno vplival na zahtevke, ki gredo na »nekaj.si«, čeprav pištkotek v osnovi ne izvira iz domene »nekaj.si«. Tak piškotek se lahko uporabi za lažne prijave ali za spremembo informacij o identiteti.

2.3.11.7 Zombie piškotek (zombie cookie)

Določeni piškotki so avtomatsko rekreriani po tem, ko jih uporabnik pobriše. Take piškotke imenujemo tudi zombie piškotki. To se izvede s programsko kodo, ki shrani vsebino piškotka na prostor, ki ni namenjen shranjevanju piškotkov (lokalni prostor namenjen za tehnologijo Flash ali prostor za shranjevanje na osnovi tehnologije HTML5). Piškotek se tipično ponovno naredi iz varne kopije, ko se ugotovi, da ni več prisoten.

2.3.11.8 Struktura piškotkov

Struktura sestoji iz:

- imena piškotka,
- vrednosti piškotka,
- časa veljavnosti piškotka,
- poti veljavnosti piškotka,
- domene veljavnosti piškotka,
- lastnosti varnosti,
- dostopnosti piškotka (ali je dostopen tudi preko programskega jezika JavaScript).

2.3.11.9 Uporaba piškotkov

Piškotki se uporabljajo za upravljanje s sejo uporabnika, personalizacijo ali sledenje. Najbolj pomemba lastnost piškotkov je dosegljivost oz. upoštevanje, podana preko atributa »Domain and Path«. Domena piškotka določi področje delovanja piškotka. Z domeno piškotek pove

brskalniku, da pošlje nazaj podatke na strežnik samo v primeru, da domena naslova ustreza poti podani v atributu »pot veljavnosti piškotka«. Če pot ni podana, je privzeta vrednost enaka vrednosti domene in poti, od koder izvira poizvedba. Tu pride do razlike med piškotkom, ki je bil poslan iz naslova »primer.si« brez atributa domene, in med piškotkom, ki je imel nastavljen atribut »pot veljavnosti piškotka« na vrednost domene »primer.si«. V prvem primeru bo piškotek posredoval vrednost strežniku samo v primeru, ko bo poizvedba imela v naslovu »primer.si«. V drugem primeru bo piškotek posredovan na strežnik tudi za vse pod domene. Primeri direktiv za nastavljanje piškotka po prijavi uporabnika na naslovu »dokumenti.primer.si« so podani v nadaljevanju (standard RFC 6265[17] določa, kako so parametri piškotkov interpretirani):

- Set-Cookie: SID=31d4d96e407aad42; Path=/marketing; Expires=Sun, 22 Jul 2014 11:21:05 GMT; Secure; HttpOnly,
- Set-Cookie: SID= 31d4d96e407aad43; Domain=.primeri.si; Path=/; Expires=Sun, 22 Jul 2014 11:21:05 GMT; HttpOnly,
- Set-Cookie: SID= 31d4d96e407aad44; Domain=primer.si; Path=/; Expires=Sun, 22 Jul 2014 11:21:05 GMT; Secure; HttpOnly.

Prvi piškotek z oznako SID 31d4d96e407aad42 nima podanega atributa za domeno, ima pa podano pot »/marketing«, ki pove brskalniku, da pri dostopu do strani, ki vsebujejo v naslovu »dokumenti.primeri.si/marketing«, uporabi piškotek. Torej za domene, ki so izpeljane iz domente zahteve. Druga dva primera posredujeta vrednost piškotka na strežnik, kadar dostopata do katerekoli poddomene znotraj ».primer.si« na vsaki poti, kot je npr. »www.primer.si/«. V drugem primeru pika podana v atributu domene ni obvezna v zadnjih različicah standarda, vendar se jo lahko poda za združljivost s standardom RFC 2109[31]. Piškotki so lahko nastavljeni samo na zgornji domeni in ne na pod domenah. Nastavljanje piškotka na domeni »www.primeri.si« iz naslova »www.diploma.si« ne bo delovalo iz varnostnih vidikov.

2.3.11.10 Alternativa piškotkom

Alternativa piškotkom je lahko v določenih primerih:

- številka IP (nekaterim uporabnikom lahko sledimo na osnovi številke IP, preko katere izvaja dostop do spletne strani),
- naslov povezave (boljša tehnika je vključitev informacij za sledenje v sam naslov povezave, kar uporabljajo kot rešitev tudi strežniški programski jeziki),

- skrita vnosna polja določenega obrazca (informacije lahko prikrijemo z uporabo skritih vnosnih polj, stran pa dobesedno zavijemo v obrazec),

- ime okna (z uporabo programskega jezika lahko danes shranimo v lastnost objekta DOM 2 – 32 MB podatkov in ta ni omejena z domeno),

- avtentikacija HTTP (od uporabnika preprosto vedno znova zahtevamo avtentikacijo, vendar brskalnik shrani vnešeno vrednost in jo posreduje ob vsaki naslednji poizvedbi).

2.3.11.11 Slabosti piškotkov

Poleg vprašanja zasebnosti imajo piškotki tudi nekaj tehničnih pomanjkljivosti. Piškotki ne morejo natančno identificirati uporabnika in so lahko predmet zlorabe, zato so pogosto v nasprotju s tehnološko arhitekturo REST (Representational State Transfer)[46]. Pomankljivosti, ki bi jih lahko pripisali piškotkom, so:

- Nenatančna identifikacija

  Če je na računalniku nameščenih več brskalnikov hkrati, imajo ločen prostor za shranjevanje piškotkov. Zato piškotek ne identificira osebe, pač pa kombinacijo uporabniškega računa, računalnika in brskalnika. Vsak uporabnik, ki uporablja več uporabniških računov ali brskalnikov, ima več naborov piškotkov. Prav tako se piškotki ne razlikujejo med večimi uporabniki, ki si delijo isti uporabniški račun, računalnik in brskalnik.

- Neskladnost med stanjem na strežniku in na odjemalcu

  Uporaba piškotov lahko pripelje do neskladnosti med stanjem na odjemalcu in stanjem shranjenim v piškotku. Če uporabnik pridobi piškotek in izbere v brskalniku gumb "Nazaj", potem stanje ni enako kot pred pridobitvijo. Poglejmo si zadevo na primeru. Imamo spletno trgovino (zasnovano na piškotkih) in uporabnik napolni košaro z enim artiklom. Ko v brskalniku uporabnik izbere gumb "Nazaj", bi morda uporabnik želel, da se artikel odstrani iz košare, vendar ravno zaradi uporabe piškotkov artikel ostane v košari. Taki primeri lahko pripeljejo do nezanesljivega delovanja, zmede in napak. Spletni razvijalec se mora zavedati načina delovanja in vgraditi prijeme, ki take pomanjkljivosti odpravljajo.

- Neskladnost pri uporabi naprav

Težava piškotkov na mobilnih napravah je, da vse mobilne naprave nimajo omogočenega dela s piškotki. Na primer Nokia podpira piškotke samo na 60% napravah, ki jih proizvaja, medtem ko Motorola podpira piškotke na 45% napravah[21]. Dodatno težavo pri uporabi piškotkov predstavljajo omrežja, ki ne omogočajo uporabe piškotkov (Verizon, Alltel, and MetroPCS) ali pa omrežja, ki simulirajo piškotke v imenu njihovih mobilnih naprav. Po svetu imamo tudi veliko število različnih variacij v brezžičnih omrežjih pri uporabi piškotkov. V Angliji podpira uporabo piškotkov v brezžičnih omrežjih kar 94% primerov vseh naprav. V Združenih državah Amerike podpira uporabo piškotkov v brezžičnih omrežjih le 47% naprav. Podpora piškotkom je večja na daljnem vzhodu, kjer brezžične naprave bolj pogosto uporabljajo za brskanje po spletnih straneh. Mobilne piškotke uporabljajo na Japonskem in tam lahko v okviru vremenske napovedi, video posnetka ali npr. televizije preko piškotkov sledimo navadam uporabnika [21].

- Nevarnost piškotkov

  Danes piškotki predstavljajo resen problem pri zagotavljanju varnosti. Kraja piškotkov je enostavna. Preko nezaščitenih identifikatorjev lahko pride do kraje seje in tisti, ki zlorablja, lahko opravlja delo v našem imenu.

2.4 Proces prijave v družabna omrežja

2.4.1 Protokoli uporabljeni na nivoju komunikacije

Ponudniki upravljanja z identitetami se v zadnjem času nagibajo k uporabi zaščitene povezave in uporabe protokola SSL/TLS. Vseeno imamo uporabo tudi v nezaščiteni komunikaciji. Primer protokola za upravljanje z identitetami, ki je danes zelo razširjen in se uporablja tako v nezaščiteni kot tudi zaščiteni povezavi, je OAuth. Ravno zaradi uporabe različnih protokolov na nivoju komunikacije sta nastali različici:

- različica OAuth 1.0 se lahko uporablja tudi pri nezaščiteni povezavi (protokol HTTP),
- različica OAuth 2.0 se uporablja izključno z uporabo protokola SSL/TLS, torej pri zaščiteni povezavi.

Poglejmo si najprej protokole na nivoju komunikacije in obliko podatkov, preko katere se izmenjujejo podatki.

2.4.1.1 Uporaba protokola SSL/TLS

Danes večina družabnih omrežij že zahteva varen način povezave, tako da se vsebina prenaša po protokolu HTTPS in uporablja varni protokol SSL/TLS. S tem je vsebina že zaščitena v primeru prisluškovanja povezavi. Vsebina spletne aplikacije pa je še vedno razvidna pri ogledu izvorne kode. Rešitev zato ponuja dodatno zaščito in je uporabna tudi v primeru uporabe varne povezave.

2.4.1.2 Uporaba protokola HTTPS

Gre za ločen protokol za prenos podatkov preko spleta. Gre za varno različico protokola HTTP. Protokol SSL poskrbi za varno povezavo med strežnikom in odjemalcem, preko katere se vsi podatki prenašajo varno. Protokol HTTPS pa je zasnovan za pošiljanje individualnih spletnih poizvedb na varen način. Protokola se zato dopolnjujeta in si nista konkurenčna. Oba protokola sta bila potrjena s strani IETF (Internet Engineering Task Force) kot standard.

2.4.2 Oblike zapisov na nivoju izmenjave podatkov

Na kratko si oglejmo oblike zapisov podatkov, ki se uporabljajo pri izmenjavi podatkov. To sta obliki XML in JSON.

2.4.2.1 Oblika podatkov XML

Oznaka XML pomeni Extensible Markup Language. Gre za označevalni jezik, ki je podoben jeziku HTML. Zasnovan je tako, da omogoča opis podatkov, ne pa njihovo vizualizacijo. Sestavljen je iz oznak (tags), ki niso preddefinirani, ampak lahko definiramo lastne oznake. Jezik XML je samo opisni in sledi standardom W3C. Lahko bi rekli, da se jezik HTML osredotoča na izgled podatkov, medtem ko se jezik XML osredotoča na pomen podatkov. Jezik XML torej ni nadomestilo za jezik HTML. Poglejmo si majhen primer dokumenta XML:

```
<posta>
<to>Janez</to>
<from>Novak</from>
<heading>Reminder</heading>
<body>Na pozabi na zagovor!</body>
</posta>
```

Zgornji dokument XML je sam po sebi opisen: imamo glavni element »posta«, ki vsebuje podatke prejemnika, naslov sporočila in vsebino sporočila. Zgornji primer tudi prikazuje uporabo lastnih oznak, kar je v primeru jezika HTML nemogoče, saj so oznake v naprej definirane.

2.4.2.2 Oblika podatkov JSON

Pregled in analiza tehnologij za serializacijo objektov[19] prikaže, da standard JSON predstavlja precejšno poenostavitev pri izmenjavi podatkov med strežnikom in odjemalcem, saj nam ni treba skrbeti za definiranje strukture podatkov pri prevzemu odgovora na tak zahtevek. Z uporabo transformacijskih funkcij dosežemo, da lahko prenašamo kompleksno vsebino predstavljeno v obliki XML. Pri izvedbi klicev na družabno omrežje največkrat pridobimo odgovor v obliki JSON. Po izvedbi programske kode v globalni kontekst brskalnika se odgovor opredeli in s tem postane rezultat dostopen preko spremenljivk. Poglejmo si primer enakovrednega objekta JSON primerljiv s primerom XML v predhodnem poglavju:

```
{
  "posta": {
    "to": "Janez",
    "from": "Novak",
    "heading": "Reminder",
    "body": "Ne pozabi na zagovor!"
  }
}
```

Če objekt priredimo spremenljivki in spremenljivko opredelimo z izvedbo v globalni kontekst, potem v programskem jeziku JavaScript lahko dostopamo do objekta preko spremenljivke. Primer psevdo kode prikazuje izvedbo v global kontekst in opredelitev spremenljivke »mto« z vrednostjo »to« v objektu JSON[50]:

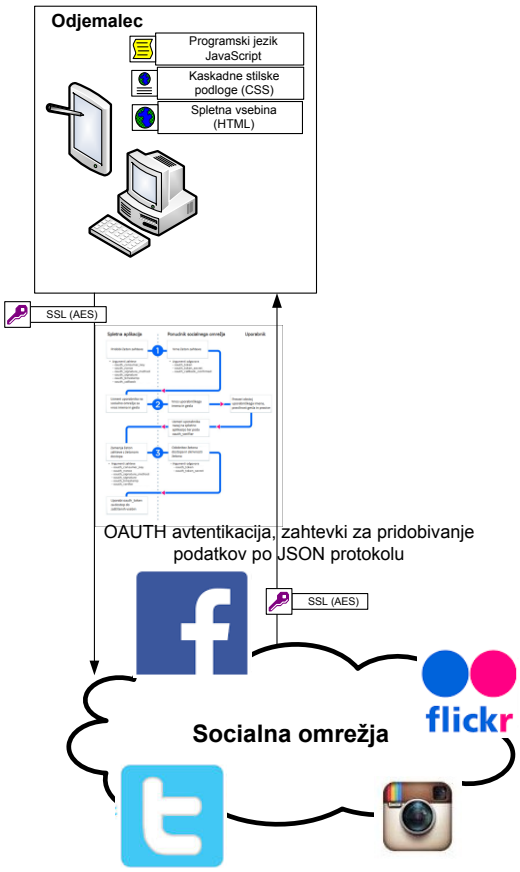
```
<scripttype="text/javascript">
  // Izvedba v globalni kontekst
  function globalEval(src) {
    if (window.execScript) window.execScript(src); else
      eval.call(null, src);
  }

  myvar = {
      "posta": {
      "to": "Janez",
      "from": "Novak",
      "heading": "Reminder",
      "body": "Ne pozabi na zagovor!"
    }
  }

  globalEval( myvar );
  setTimeout( function() {
    mto = myvar[0].posta.to;
  }, 200 );
</script>
```

Prenos iz odjemalca do strežnika lahko opravimo tako, da vsebino spremenljivke z algoritmom spremenimo v obliko XML na strani odjemalca in nato vsebino preko

asinhronega zahtevka posredujemo na strežnik, da ga zapiše v podatkovno bazo Oracle, stolpec tipa XMLTYPE. Podatkovna baza Oracle namreč prinaša največ prednosti pri obravnavanju tako strukturiranih podatkov.

2.4.3 Načini prijave pri zunanjih ponudnikih upravljanja z identitetami

Prvotni namen pri izgradnji spletne aplikacije je bil povezljivost z družabnimi omrežji oziroma izgradnja aplikacije za podporo družabnega omrežja Facebook in možnost razšititve na druga družabna omrežja, kot je Flickr. Z izvedbo prijave v družabno omrežje spletna stran na strani odjemalca pridobi pravice za dostop do uporabnikovih privatnih elementov, kot so prijatelji, albumi, osebni podatki (primer družabnega omrežja Facebook). Nabor podatkov, ki jih spletna stran vidi, je odvisen od pravic, ki jih navedemo pri vključitvi v družabno omrežje. Če uporabnik seznam pravic, ki jih zahtevamo, potrdi, pridobi spletna aplikacija pravice, ki so bile navedene. Združitev več družabnih omrežij oziroma možnost, da se uporabnik v spletni aplikaciji prijavi v več družabnih omrežij hkrati, ponuja postopek prijave, ki uporablja protokol OAuth. Slika 6 prikazuje shemo povezave med uporabnikom spletne aplikacije in družabnimi omrežji.



Slika 6: Shema povezave med odjemalcem in družabnim omrežjem

Uporaba vmesnikov družabnih omrežij v zadnjem času že zahteva uporabo varnega protokola SSL/TLS in protokola HTTPS. Povezava preko standardnega komunikacijskega kanala ni več povsod mogoča in ta možnost se umika. To sicer ne predstavlja omejitev, vendar je treba pridobiti strežniški certifikat, ki ga je treba kupiti.

V nadaljevanju si poglejmo glavne protokole za upravljanje z identitetami v družabnih omrežjih.

2.4.3.1 Različica OAuth 1.0

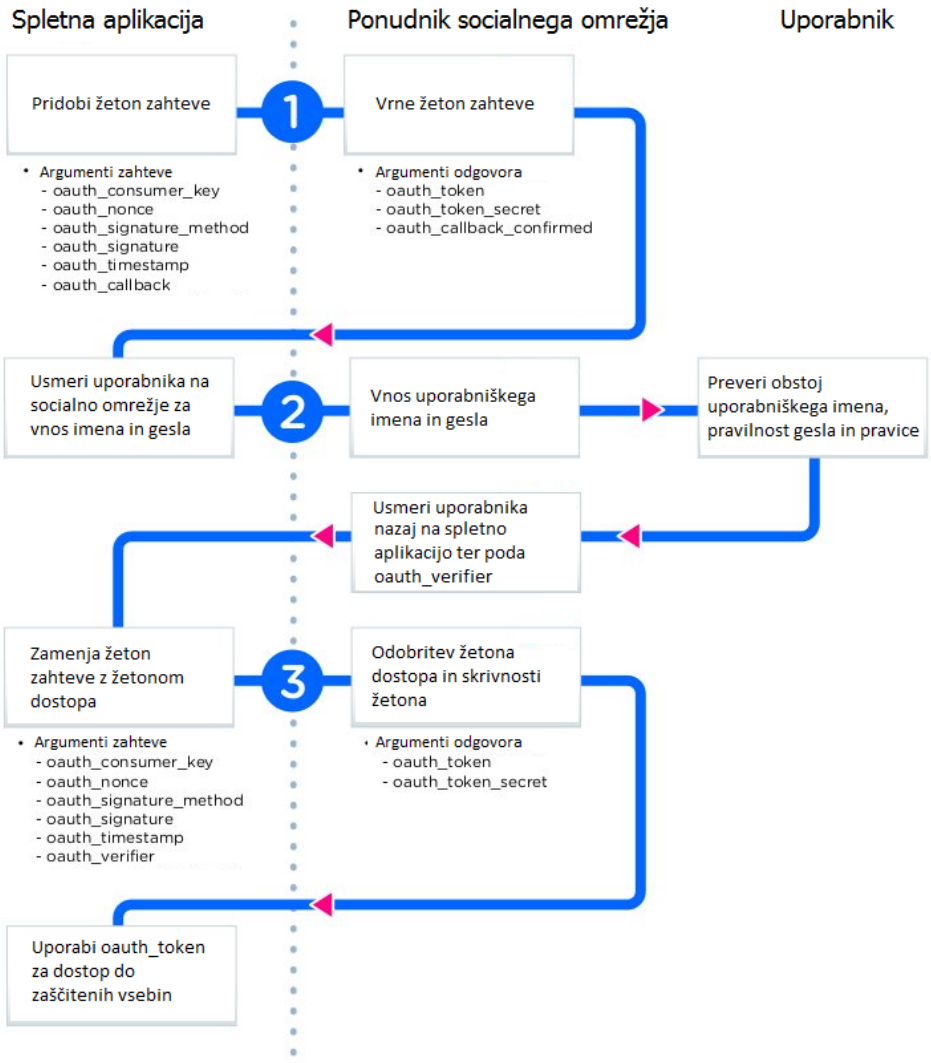
OAuth je odprtokoden protokol definiran v standardu RFC 5849[18], ki omogoča aplikacijam avtentikacijo uporabnikov in interakcijo z družabnim omrežjem v uporabnikovem imenu. Podatki uporabnika se prenašajo na varen način in zaupni podatki identitete niso izpostavljeni. V družabnem omrežju Flickr, ki uporablja različico OAuth 1.0A, se protokol uporablja, da preveri, ali aplikacija, ki uporablja storitve Flickr v uporabnikovem imenu, ima pravice, ki jih je uporabnik omogočil. Potek OAuth ime 3 glavne korake:

- pridobitev žetona zahteve,

- pridobitev uporabnikove avtorizacije,
- menjava žetona zahteve z žetonom dostopa.

Primer uporabe OAuth z družabnim omrežjem Flickr

Slika 7 prikazuje potek prijave z izmenjavo žetonov v družabno omrežje Flickr z uporabo protokola OAuth.



Slika 7: Prijava v družabno omrežje z uporabo protokola OAuth

Pomembni koraki pri izvedbi avtentikacije so:

1. Podpisovanje poizvedb

Vse poizvedbe do vmesnika morajo biti podpisane z uporabo HMAC-SHA1 šifriranja, ki šifrira po standardu RFC 2104[30]. Razlog je, da se poizvedbe izvajajo na nezaščiten način oz. z uporabo protokola HTTP. Šifriranje izvedemo z naslednjimi koraki. Najprej pridobimo niz, ki je sestavljen iz besede HTTP, povezave in iz vseh argumentov, ki morajo biti urejeni po imenu in ločeni z znakom »&«. V okviru nalaganja spletne aplikacije se praviloma izvede prijava v družabna omrežja pred vmesnikom za stiskanje in šifriranje, saj se v okviru prijave na družabno omrežje z uporabo protokola OAuth izvaja večkratna preusmeritev in večkratno nalaganje kompleksne logike bi bilo časovno potratno. Ključ je sestavljen iz uporabnikove skrivnosti, znaka '&' in žetona. Poglejmo si zadevo na primeru. Imamo osnovno povezavo:

```
https://www.flickr.com/services/oauth/request_token
?oauth_nonce=89601180
&oauth_timestamp=1305583298
&oauth_consumer_key=653e7a6ecc1d528c516cc8f92cf98611
&oauth_signature_method=HMAC-SHA1
&oauth_version=1.0
&oauth_callback=http%3A%2F%2Fwww.primer.si
```

Iz povezave dobimo niz:

```
GET&https%3A%2F%2Fwww.flickr.com%2Fservices%2Foauth%2Frequest_token&oauth_ca
llback%3Dhttp%253A%252F%252Fwww.primer.si%26oauth_consumer_key%3D653e7a6ecc1
d528c516cc8f92cf98611%26oauth_nonce%3D95613465%26oauth_signature_method%3DHM
AC-SHA1%26oauth_timestamp%3D1305586162%26oauth_version%3D1.0
```

Niz generira naslednji podpis:

```
7w18YS2bONDPL%2FzgyzP5XTr5af4%3D
```

2. Pridobitev žetona za zahtevo (poizvedbo)

Najprej spletna stran pridobi žeton zahtevka, po izvedbi uspešne prijave v družabno omrežje pa žeton dostopa. Uporaba žetona dostopa je enakovredna prijavi v družabno omrežje. Žeton shranimo v piškot v brskalnik in vnovična prijava uporabnika ni več zahtevana. Izmenjava žetonov se izvaja s podpisovanjem zahtevka JSON. Pridobitev žetona za poizvedbo se izvede preko povezave vmesnika:

```
https://www.flickr.com/services/oauth/request_token
```

Prvi korak za izvedbo avtorizacije uporabnika je pridobitev žetona za dostop preko ključa stranke. To je začasni žeton, ki bo uporabljen za izvedbo avtentikacije

uporabnika v aplikacijo. Ta žeton v kombinaciji s skrivnostjo bo kasneje zamenjan z žetonom dostopa. Primer pridobitve žetona za poizvedbo:

```
https://www.flickr.com/services/oauth/request_token
?oauth_nonce=95613465
&oauth_timestamp=1305586162
&oauth_consumer_key=653e7a6ecc1d528c516cc8f92cf98611
&oauth_signature_method=HMAC-SHA1
&oauth_version=1.0
&oauth_signature=7w18YS2bONDPL%2FzgyzP5XTr5af4%3D
&oauth_callback=http%3A%2F%2Fwww.primer.si
```

Flickr vrne odgovor v obliki:

```
oauth_callback_confirmed=true
&oauth_token=72157626737672178-022bbd2f4c2f3432
&oauth_token_secret=fccb68c4e6103197
```

3. Avtorizacija uporabnika

Avtorizacija uporabnika se izvede preko povezave vmesnika:

```
https://www.flickr.com/services/oauth/authorize
```

Po pridobitvi žetona poizvedbe mora spletna aplikacija izvesti preusmeritev na stran Flickr za avtorizacijo, kjer uporabnik potrdi dovoljenje spletni aplikaciji za dostop do uporabnikovih podatkov. Stran za avtorizacijo ponudi uporabniku seznam pravic, ki jih aplikacija zahteva od uporabnika. Dodatno lahko podamo parameter "perms=", s katerim pridobimo dovoljenje za branje, pisanje in/ali brisanje. Če je parameter podan, potem se privzete pravice (podane na spletni strani ponudnika storitve) ne upoštevajo, pač pa se upoštevajo pravice podane v parametru. Po izvedbi avtorizacije ponudnik storitve izvede preusmeritev nazaj na našo spletno aplikacijo, pri čemer v odgovoru poda parameter "oauth_callback", ki je podan v žetonu poizvedbe. Primer povezave za izvedbo avtorizacije:

```
https://www.flickr.com/services/oauth/authorize
?oauth_token=72157626737672178-022bbd2f4c2f3432
```

Primer povezave, na katero ponudnik preusmeri uporabnika po uspešni izvedbi avtorizacije, je:

```
http://www.example.com/?oauth_token=72157626737672178-022bbd2f4c2f3432
&oauth_verifier=5d1b96a26b494074
```

4. Izmenjava žetona poizvedbe za žeton dostopa

   Ko uporabnik avtorizira aplikacijo, lahko izmenjamo začasni žeton poizvedbe z žetonom dostopa. Ta žeton spletna aplikacija shrani v piškotek za nadaljnje dostope na avtoriziran način. Naslov povezave vmesnika za pridobitev žetona dostopa je:

   ```
   https://www.flickr.com/services/oauth/access_token
   ```

   Primer poizvedbe, s katero pridobimo žeton za dostop:

   ```
   https://www.flickr.com/services/oauth/access_token
   ?oauth_nonce=37026218
   &oauth_timestamp=1305586309
   &oauth_verifier=5d1b96a26b494074
   &oauth_consumer_key=653e7a6ecc1d528c516cc8f92cf98611
   &oauth_signature_method=HMAC-SHA1
   &oauth_version=1.0
   &oauth_token=72157626737672178-022bbd2f4c2f3432
   &oauth_signature=UD9TGXzrvLIb0Ar5ynqvzatM58U%3D
   ```

   Odgovor je v obliki:

   ```
   fullname=Dominik%20Lebar
   &oauth_token=72157626318069415-087bfc7b5816092c
   &oauth_token_secret=a202d1f853ec69de
   &user_nsid=21207597%40N07
   &username=dominiklebar
   ```

5. Uporaba vmesnika z OAuth po pridobitvi žetona za dostop

   Po pridobitvi žetona za dostop lahko izvajamo poizvedbe preko vmesnika za uporabo storitev. Primer klica metode *flickr.test.login* z uporabo žetona za dostop:

   ```
   https://api.flickr.com/services/rest?nojsoncallback=1
   &oauth_nonce=84354935
   &format=json
   &oauth_consumer_key=653e7a6ecc1d528c516cc8f92cf98611
   &oauth_timestamp=1305583871
   &oauth_signature_method=HMAC-SHA1
   &oauth_version=1.0
   &oauth_token=72157626318069415-087bfc7b5816092c
   &oauth_signature=dh3pEH0Xk1qILr82HyhOsxRv1XA%3D
   &method=flickr.test.login
   ```

   Primer odgovora v obliki JSON v našem primeru:

```
{"user":
  {
    "id":"21207597@N07",
    "username":
    {
      "_content":"dominiklebar"
    }
  },
  "stat":"ok"
}
```

2.4.3.2 Različica OAuth 2.0

Največja slabost prve različice je kompleksnost pri izvedbi kriptografije. Določena družabna omrežja omogočajo uporabo enostavne kode za hitro uporabo določenih storitev. Pri uporabi OAuth je razvijalec prisiljen uporabiti eno izmed obstoječih knjižnic, namesto da poda ukaz v eni vrstici z uporabo cURL. Pri protokolu OAuth različica 2 ni prisotno podpisovanje poizvedb, vendar obstaja zahteva, da gredo določene poizvedbe preko varnega protokola SSL. Protokol OAuth 2.0 ni dokončen in se še vedno razvija, medtem ko je različica 1.0a končna. Protokol omogoča nekaj novih potekov oziroma izvedb avtentikacije:

- potek za odjemalca (brskalnik),
- potek za strežnik (za odjemalce, ki so del spletnega strežnika – podoben potek kot OAuth različica 1),
- potek za naprave (za naprave z omejenimi zmožnostmi),
- uporabniško ime in geslo (za uporabo, kjer strežnik zaupa odjemalcem, čeprav je še vedno nezaželjeno, da uporabnik shranjuje uporabniško ime in geslo),
- potek odjemalca z uporabo gesla (uporabnik uporabi uporabniško ime in geslo, da pridobi žeton),
- potek preko protokola SAML (odjemalec posreduje dokument po protokolu SAML, da pridobi žeton).

Eden izmed glavnih uporabnikov protokola OAuth 2.0 so Facebook Graph API, Google in Microsoft.

2.4.3.3 OpenID

Protokol OpenID dovoli uporabo obstoječega uporabniškega računa za prijavo v več spletnih strani brez potrebe po kreiranju novih uporabniških računov in / ali gesel. Uporabnik lahko

poveže informacije, kot so ime in elektronski naslov v OpenID in jih deli z drugimi spletnimi stranmi. S protokolom OpenID lahko kontroliramo, koliko informacij želimo deliti s spletnimi stranmi, ki jih obiskujemo. Z uporabo tega protokola je geslo posredovano samo ponudniku identitete. Ta potrdi identiteto spletni strani, ki jo obiskujemo. Na ta način nam ni treba skrbeti za pomanjkljivo varnost, ki bi ogrozila uporabnikovo identiteto. Protokol je danes v razmahu, saj imamo več kot bilijon uporabiških računov, ki uporabljajo enega izmed ponudnikov protokola. Tudi rast spletnih strani s podporo avtentikaciji OpenID je velika. Danes je veliko organizacij, ki so ali ponudniki protokola ali ponudniki identitet (identity provider), ki omogočajo avtentikacijo s protokolom OpenID. Med njimi so npr. Google, Facebook, Yahoo!, Microsoft, AOL, MySpace, Sears, Universal Music Group, France Telecom, Novell, Sun, Telecom Italia ipd. Protokol OpenID (OID) je bil kreiran v letu 2005, da bi obšel težave, ki niso bile enostavno rešljive s strani obstoječih tehnologij za upravljanje z identitetami. Je decentraliziran odprt standard neprofitne organizacije OpenID Foundation, ki omogoča uporabnikom, da se avtenticirajo preko tretje stranke in omogoča konsolidacijo digitalnih identitet. Uporabnik se lahko prijavi v več nepovezanih spletnih mest, brez nuje po vedno novi registraciji. Nekateri ponudniki OpenID so razvili tudi lastno shemo za upravljanje z identitetami, drugi so uporabo OpenID opustili. Tak primer je podjetje Facebook, ki danes uporablja Facebook Connect. Danes lahko vsak uporabi OpenID ali postane ponudnik tega protokola, brez da bi bilo treba izvesti registracijo ali da bi bilo treba dobiti dovoljene za uporabo.

Razširitev standarda OpenId (OpenID Attribute Exchange) vpeljuje možnost prenosa uporabnikovih atributov, kot so spol in ime ponudnika identitete. Prva različica namreč ne nudi ponudniku spletne aplikacije informacije o elektronski pošti ali imenu. Ne vemo niti, ali gre za človeka ali robota.

Protokol ni odvisen od overitelja potrdil, da zagotovi avtentikacijo uporabniške identitete.

2.4.3.4 OpenID Connect (OIDC)

Protokol, objavljen v letu 2014 s strani organizacije OpenID Foundation, predstavlja tretjo generacijo tehnologije OpenID. Gre za nivo avtentikacije, ki je implementiran nad vmesnikom OAuth 2.0 in prinaša prijaznejši način izvajanja preko vmesnika ter izvajanje istih funkionalnosti kot z uporabo protokola OpenID 2.0. Največja razlika v primerjavi z OpenID 2.0 je v vmesniku, ki nudi lažjo vgradnjo v mobilne aplikacije. Vmesnik OpenID Connect ima tudi možnost uporabe mehanizmov za podpisovanje in šifriranje. V primeru povezovanja med OAuth 1.0a in OpenID 2.0 je potrebna razširitev, med tem ko je v OpenID Connect že vgrajena podpora protokola OAuth 2.0 v sam protokol.

2.4.3.5 Facebook Connect

Facebook Connect je sistem za enotno prijavo, ki omogoči uporabnikom interakcijo na drugih spletnih straneh preko njihovega računa Facebook. Protokol Facebook Connect olajša dostop do spletne aplikacije na avtenticiran način z uporabo vmesnika tako, da omogoči prijavo v tretje spletne strani, aplikacije, mobilne naprave in igre z uporabo identitete Facebook. Med tem ko je uporabnik prijavljen v sistem Facebook, lahko v sklopu spletne aplikacije, ki uporablja avtentikacijo Facebook Connect, izvaja storitve Facebook, kot so povezovanje s prijatelji, objavljanje informacij na uporabniškem profilu in podobno. Razvijalec lahko tako vključuje storitve Facebook v lastno spletno aplikacijo in daje objave v skupno rabo. Protokol Facebook Connect je še v fazi razvoja in uporablja vmesnik Facebook Platform. Ena izmed možnosti uporabe Facebook Platform je z uporabo knjižnice JavaScript. Koraki za vključitev so naslednji:

- preverimo status prijave,
- v primeru, da uporabnik ni prijavljen, ga preusmerimo na stran za prijavo, kjer od uporabnika zahtevamo pravice,
- preverimo identiteto,
- pridobimo in shranimo žeton dostopa,
- izvajamo klice vmesnika s podanim žetonom dostopa,
- uporabniku ponudimo možnost odjave.

Poglavje 4 prikazuje primer, ki sledi opisanim korakom.

2.4.4 Primerjava avtentikacije prek različnih zunanjih ponudnikov

2.4.4.1 Primerjava med OpenID in Single Sign On

Primerjava[8] med OpenID in Single sign on (SSO) strežnikom pokaže, da si nista enakovredna. OpenID ni SSO strežnik. Njegov namen je malo drugačen. Poglejmo si na primeru: imamo dva strežnika, dve spletni aplikaciji, vsaka zahteva od uporabnika, da se prijavi. Pri uporabi ne bi radi, da se mora uporabnik v spletno aplikacijo prijaviti dva krat, pač pa, ne glede na katero spletno stran gre najprej, ga spletna stran preusmeri na stran za prijavo in od tega trenutka naprej ima dostop do obeh spletnih strani. V nadaljevanju si najprej poglejmo kakšne imamo omejitve pri uporabi piškotkov in kdaj potrebujemo uvedbo enotnega strežnika (single sign on). Potem si bomo pogledali kam v tej zgodbi sodi OpenID.

Enostaven primer: strežnik v skupni rabi (shared host)

Poglejmo si primer, če sta spletni aplikaciji na istem strežniku. Obe spletni aplikaciji lahko uporabljata isti podatkovni strežnik, ki lahko hrani uporabniško ime in geslo v piškotku. V piškotku lahko naslovimo enako pot, ki se sklicuje na sam strežnik. Lahko uporabimo tudi ločena strežnika in z uporabo "load-balancer" ali "reverse proxy" dosežemo, da so poizvedbe preusmerjene na pravi strežnik.

```
http://example.com/secrets/
http://example.com/mysteries/
```

Enostaven primer: domena v skupni rabi (shared domain)

V tem primeru se strežnika lahko nahajata geografsko na ločenih mestih in deljenje istega imena za strežnik ne pride v upoštev. Tu piškotek še vedno lahko nastavimo tako, da ga spletni aplikaciji delita. V našem primeru

```
http://secrets.example.com/
http://mysteries.example.com/
```

nastavimo lastnost domen na ".example.com". Rešitev ne pride v poštev, če strežnika ne uporabljata iste domene.

Napreden primer: strežnik za enotno prijavo (single-sign-on server)

Tu spletni aplikaciji nimata skupnega prednika – iste domene in piškotka ne moremo deliti med spletnima aplikacijama.

```
http://secrets.com/
http://mysteries.com/
```

Piškotka ne moremo nastaviti na način, da bi bil uporabljen pri obeh domenah. V tem primeru potrebujemo tretji strežnik, katerega namen je beležiti sledi, kdo je kje prijavljen. Ko uporabnik obišče prvo spletno stran in piškotek ni nastavljen, preveri preko SSO strežnika, ali je uporabnik že prijavljen in v tem primeru na tih način kreira piškotek, ki pove spletni aplikaciji, da je uporabnik prijavljen.

Uporabniška baza v skupni rabi (shared user database)

Integracija

Zgoraj smo prikazali pomemben korak v procesu prijave uporabnika v spletno aplikacijo. Namen koraka je, da spletna stran preveri, ali je neka poizvedba veljavna ali neveljavna. V

zgornjem primeru lahko spletni aplikaciji uporabljata ločene podatkovne strežnike, ki hranijo informacije o uporabnikih. Iz tega sledi prvi korak pri združevanju spletnih aplikacij, in sicer na način, da uporabljata isti podatkovni strežnik za informacije o uporabnikih.

Priključitvena avtentikacija (pluggable authentication)

Če je spletna aplikacija produkt, ki bo nameščen na strežnikih stranke, potem bo nameščena v okolju, kjer ima stranka že svojo podatkovno bazo uporabnikov. Kopiranje podatkov uporabnikov v podatkovni strežnik, ki ga uporablja aplikacija, bi bilo zamudno in bi dopuščalo veliko napak. Boljši pristop se zdi, da spletna aplikacija uporabi svojo shemo za avtentikacijo. V ASP.NET spletnih aplikacijah je možen enostaven pristop z uporabo vmesnika, ki enkapsulira znanje za izvedbo avtentikacije. S tako zasnovano aplikacijo, ki pa je možna le, če je uporabljena izključno ASP.NET tehnologija za razvoj spletnih aplikacij, lahko dosežemo lažjo integracijo in lažje nadgrajevanje. V primeru, da imamo različne tehnologije za spletne aplikacije, je rešitev težja.

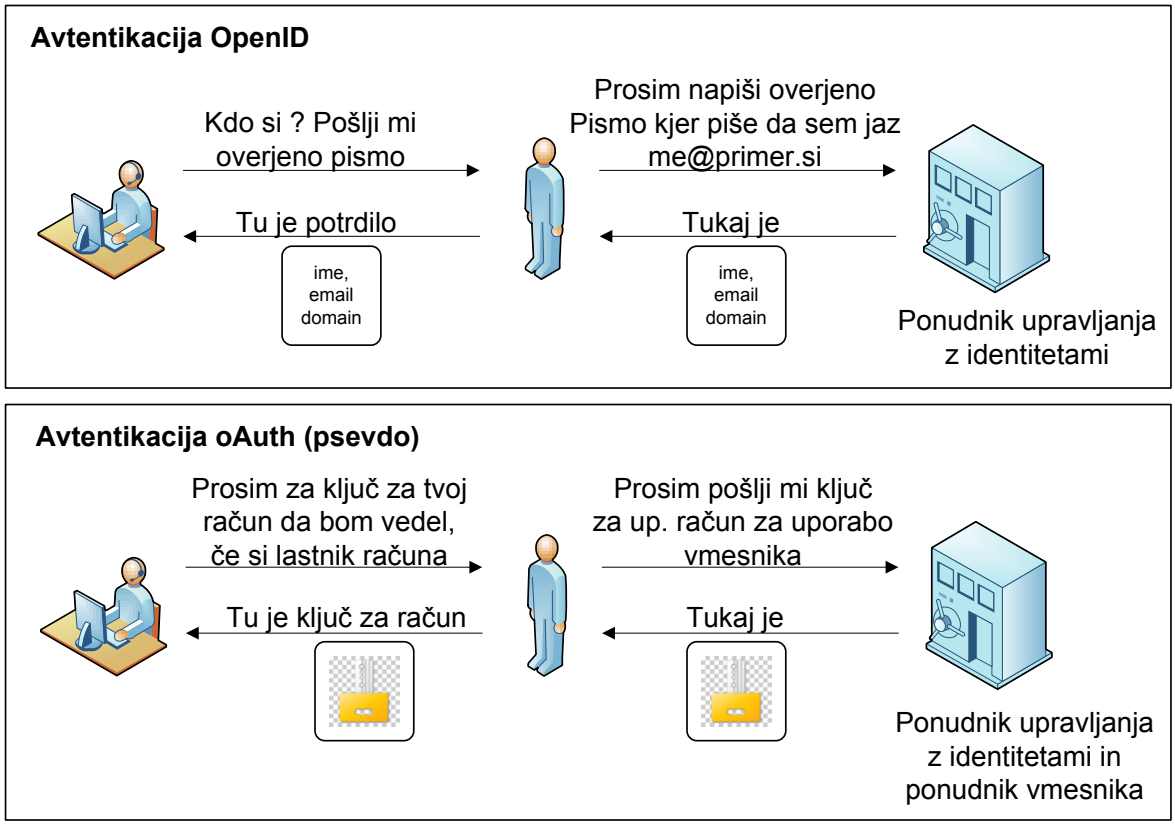
Kam sodi OpenID?

OpenID strežnik glede na namen in možnosti ni enakovreden SSO strežnikom. Bistvena razlika je, da bi v primeru vgradnje OpenID v obe spletni aplikaciji, od uporabnika zahtevali posebej avtentikacijo za obe spletni strani. Kar OpenID odpravlja, je zahteva po ponovni avtentikaciji pri obisku iste spletne strani. OpenID ustreza po izvedbi "Shared User Database" in "Pluggable Authentication". Namesto izgradnje vmesnika v ASP.NET spletni aplikaciji vgradimo podporo za OpenID v strežnik za avtentikacijo. Nekateri strežniki za avtentikacijo že podpirajo OpenID izvedbo avtentikacije. Prednosti:

- dodajanje podpore OpenID za izvedbo avtentikacije omogoča njeno uporabo tudi drugim spletnim aplikacijam, ki imajo OpenID podporo vgrajeno,
- strežnik za OpenID je lahko ločen od strežnika aplikacije in omogoča, da spletna aplikacija gostuje drugje, strežnik za izvajanja avtentikacije pa v njihovem omrežju,
- protokol OpenID omogoča, da lahko uporabimo več strežnikov za avtentikacijo skupaj, kar pride prav, ko imamo nezdružljive skupine ljudi (npr. v primeru združevanja podjetij).

2.4.4.2 Primerjava med avtentikacijo OpenID in OAuth

Slika 8 prikazuje diagram, ki prikazuje razlike med avtentikacijo OpenID in OAuth.



Slika 8: Primerjava med avtentikacijo OpenID in psevdo avtentikacijo OAuth

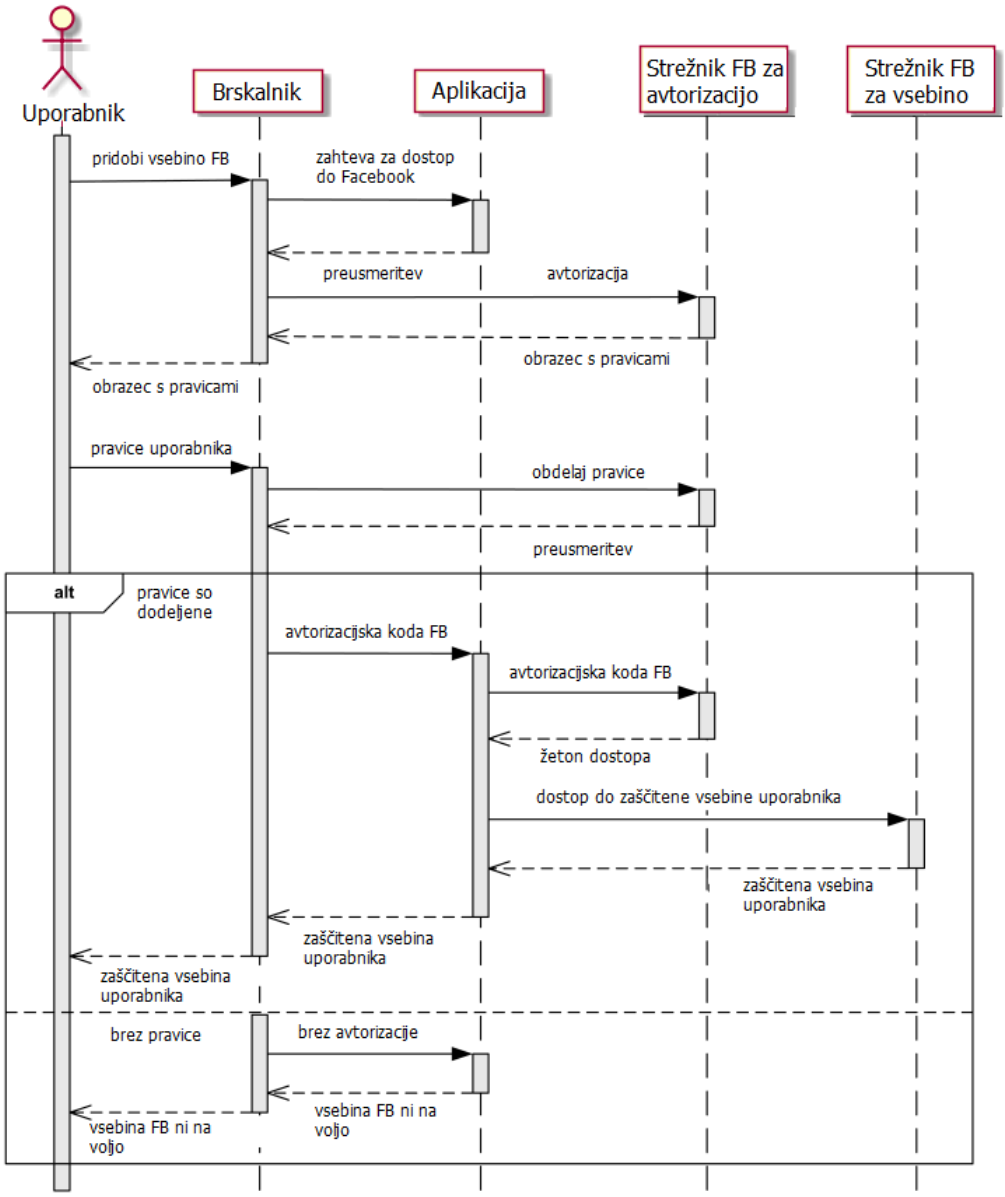
Ko se proces pri OpenID začne, aplikacija vpraša uporabnika po identiteti in uporabnik izvede avtentikacijo preko preusmeritve na naslov OpenID. V primeru uporabe protokola OAuth aplikacija posebej zahteva žeton za omejen dostop do vmesnika (ključ uporabniškega računa) v imenu uporabnika (kar posebej določi tudi pravice, ki jih aplikacija zahteva) in ne zahteva od uporabnika vnosa podatkov za prijavo. Če je uporabnik dovolil aplikaciji dostop, ta lahko pridobi uporabniško identiteto za vzpostavitev povezave z osebnim profilom preko vmesnika. V obeh primerih dostop do ponudnika uporabniške identitete vključuje avtentikacijo preko ponudnika storitve, razen če je seja že aktivna. Rezultat v primeru uporabe OpenID avtentikacije je, da aplikacija dovoli uporabniku dostop, ker zaupa ponudniku uporabniške identitete OpenID. Rezultat v primeru uporabe OAuth je, da vmesnik ponudnika dovoli aplikaciji dostop, ker zaupa lastnim ključem uporabniških računov.

Poglavje 3 Načrt rešitve, ki uporablja OAuth in Facebook Connect

Osnovni namen je bil preučitev možnosti in zahtevnosti pri izdelavi spletne aplikacije, ki vključuje storitve upravljanja z identitetami in druge storitve družabnega omrežja. Želeli smo zgraditi aplikacijo, ki bo uporabljala avtentikacijo v družabna omrežja, ter oceniti, kako se to obnese v praksi. Zanimalo nas je tudi kako težko je razviti tako spletno aplikacijo in ali lahko naredimo vmesnik, ki bi poenostavil uporabo.

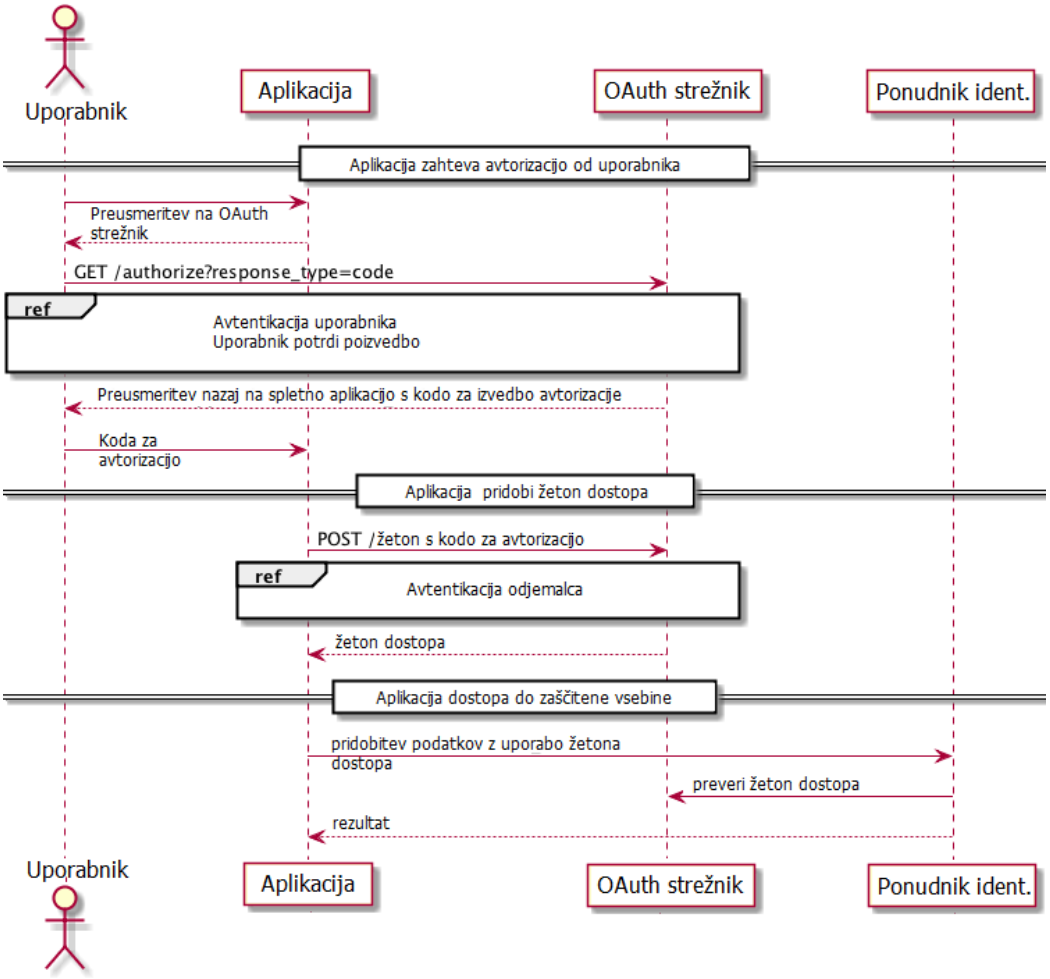
Pri načrtu spletne aplikacije smo imeli naslednje osnovne zahteve:

1. Aplikacija, ki podpira družabno omrežje Facebook. Slika 9 prikazuje shemo poteka avtentikacije pri uporabi protokola Facebook Connect in uporabo vmesnika.



Slika 9: UML diagram Facebook connect prijave

2. Aplikacija ima možnost vključitve večjega nabora družabnih omrežij z uporabo protokola OAuth. Slika 10 prikazuje shemo poteka avtentikacije pri uporabi protokola OAuth in uporabo vmesnika po avtentikaciji.



Slika 10: UML diagram OAuth prijave

3. Aplikacija zagotavlja varnost pri komunikaciji z uporabo primernih protokolov. V poglavju 4.5 smo prikazali možnost zaščite v komunikaciji, kadar ta ni na voljo. Vendar tu ni poudarek na zagotavljanju celovite zaščite, pač pa preprečevanje zlorab, ki so enostavne za izvedbo kot je na primer prisluškovanje nezaščiteni komunikaciji.

4. Aplikacija nudi uporabniku prijazno uporabniško izkušnjo.

5. Aplikacija nudi uporabniku dodatne storitve, ki jih ponudniki družabnih omrežij omogočajo.

6. Rešitev omogoča varno shranjevanje v strukturirani obliki XML in hitro iskanje podatkov v podatkovnem strežniku. Podatke zaradi obdelave na strežniku pred posredovanjem na strežniku spremenimo v obliko XML. Tako so podatki najprimernejši za takojšnjo obdelavo. V poglavju 4.4 je prikazan način varnega shranjevanja na strežniku in možnost

zapisa strukturiranih dokumentov tipa XML v podatkovni strežnik. Prikazana je ena izmed možnosti hitrega iskanja po takih dokumentih.

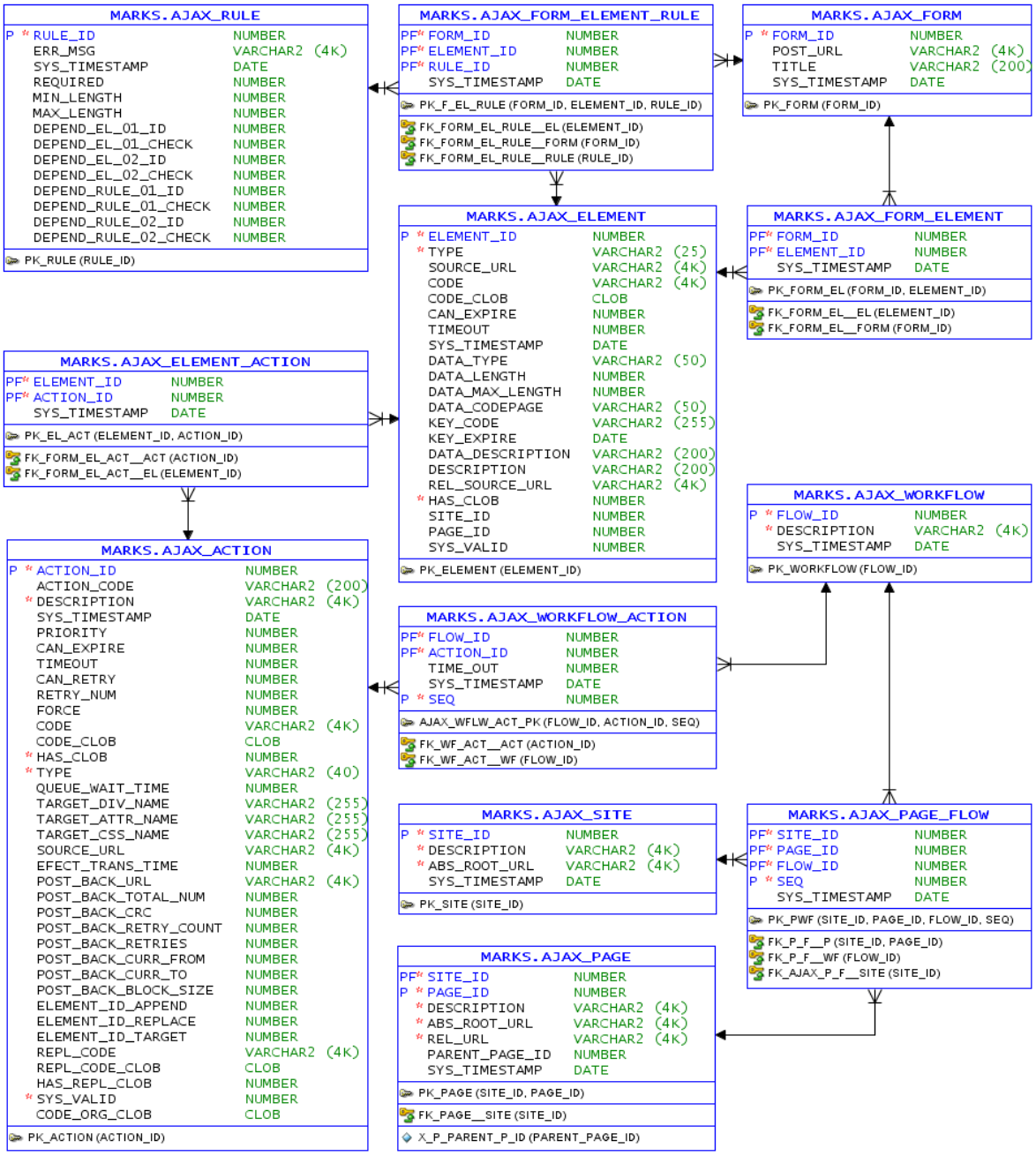
7. Podpora tabličnim računalnikom podjetja Apple. Podpora tabličnim računalnikom zahteva premostitev nekaterih omejitev, ki jih imajo brskalniki na mobilnih in tabličnih napravah. Za premostitev teh težav in večjo kontrolo pri izvajanju programske kode smo razvili »upravljalca spletne strani«. Za namen ponovne uporabe posameznih knjižnic in odpravljanje težave, povezane z odvisnostjo med knjižnicami, smo razvili podatkovni model, ki ga upravljalec strani spletne aplikacije uporablja. Model omogoča opis poteka nalaganja posameznih delov programske kode, ki sestavljajo spletno stran. Na ta način je možna ponovna vgradnja in uporaba istih vtičnikov tudi v drugih spletnih aplikacijah in to le z nekaj vpisi v tabele, ki predstavljajo potek nalaganja spletnih elementov in vsebino posameznih spletnih elementov. Orodje, ki smo ga uporabili za izdelavo relacijskega podatkovnega modela in projektne dokumentacije, je Oracle SQL Developer Data Modeler. Brez ustrezne projektne dokumentacije je razvijalcu težko razumeti, kakšni so objekti na podatkovnem strežniku in kako se med seboj povezani. Tabela 2 zaradi angleškega poimenovanja tabel opisuje pomen posameznih tabel, ki nastopajo pri upravljalcu vsebine spletne aplikacije:

Ime tabele	**Opis**
AJAX_SITE	Tabela predstavlja seznam spletnih aplikacij. Vsaka aplikacija lahko uporabi že izdelane elemente, kot tudi celotne poteke izvajanja nalaganja programske kode in vsebine spletne strani.
AJAX_PAGE	Tabela predstavlja seznam strani v posamezni spletni aplikaciji.
AJAX_WORKFLOW	Tabela predstavlja potek izvajanja.
AJAX_ELEMENT	Tabela predstavlja posamezne elemente spletne strani. Lahko gre za izvorno kodo ali samo vsebinski objekt, ki ga prikazujemo v brskalniku.
AJAX_FORM	Tabela predstavlja obrazce. Te lahko ponovno uporabimo v različnih spletnih aplikacijah (npr. obrazec za posredovanje kontaktnih informacij).
AJAX_RULE	Tabela poslovnih pravil, ki se izvajajo na obrazcih.
AJAX_ACTION	Tabela opisuje izvedbene korake oziroma opravila, kot so nalaganje programske kode, spreminjanje lastnosti objekta, ki ga prikazujemo v brskalniku, prenos podatkov ipd.
AJAX_PAGE_FLOW	Relacijska tabela vsebuje poteke, ki se na strani izvajajo.

AJAX_WORKFLOW_ACTION	Relacijska tabela vsebuje podatke, ki opisujejo, iz katerih opravil je sestavljen potek.
AJAX_ELEMENT_ACTION	Relacijska tabela predstavlja opravila, ki jih lahko izvaja posamezni element.
AJAX_FORM_ELEMENT	Relacijska tabela predstavlja elemente na obrazcu.
AJAX_FORM_ELEMENT_RULE	Relacijska tabela povezuje elemente na obrazcu s poslovnimi pravili.

Tabela 2: Pomen tabel, ki nastopajo pri upravljalcu vsebine strani spletne aplikacije

Slika 11 prikazuje relacijski model, uporabljen pri upravljalcu vsebine spletne strani:



Slika 11: Relacijski podatkovni model za dinamično posredovanje vsebine

Poglavje 4 Opis rešitve

Izvedba rešitve zajema uporabo programskega jezika JavaScript na strani odjemalca in programski jezik PHP na strani strežnika. Spletna aplikacija zagotavlja povezljivost z družabnimi omrežji, pri čemer je zagotovljeno varno povezovanje z uporabo standardnih protokolov, možnost dodatne zaščite v primeru, da varni protokoli niso na voljo in varno hranjenje podatkov na strežniku. Ciljni trg pri razvoju je zajemal moderne naprave, kot so mobilne naprave in tablični računalniki. Rešitev je že uporabljena v celozaslonski spletni aplikacijiin je omogočila vključitev več družabnih omrežij hkrati z izvedbo prijave po protokolu OAuth. Z izbiro različice podatkovnega strežnika Oracle 12c smo poskrbeli tudi za varnost pri shranjevanju podatkov. Različica podatkovnega strežnika Oracle 12c omogoča v rešitvi:

- varno shranjevanje podatkov v podatkovnem strežniku,
- napreden podatkovni prostor za shranjevanje dokumentov v strukturirani obliki XML,
- hitro iskanje po strukturiranih in nestrukturiranih podatkih.

Izdelek Oracle Cloud Control je uporabljen kot nadzorni sistem nad delovanjem podatkovnega in spletnega strežnika in kot sam spletni strežnik. Poglejmo tehnologije, uporabljene pri razvoju.

4.1 Tehnologije uporabljene pri razvoju

Pri razvoju vmesnika in spletne aplikacije je bil uporabljen koncept spleta druge generacije[37]. Pri konceptu spleta druge generacije govorimo o načinu uporabe storitev spleta in zajema dinamično ter interaktivno nudenje spletne vsebine končnemu uporabniku. Uporabniku ni potrebno osveževanje celotne spletne strani, pač pa se deli spletne strani lahko dinamično spreminjajo s časom. Pri razvoju spletne aplikacije so uporabljene naslednje tehnologije:

- podpora avtentikaciji Facebook Connect in OAuth,

- asinhron prenos podatkov, imenovan AJAX, ki uporablja obliko XML ali JSON za izmenjavo podatkov med odjemalcem in strežnikom,
- napredno stiskanje in šifriranje podatkov na podatkovnem strežniku,
- napredne strukture za hranjenje in obdelavo podatkov v obliki XML,
- kontekstni indeks za iskanje znotraj strukturiranih dokumentov XML na podatkovnem strežniku,
- programski jezik PHP na spletnem strežniku za razvoj dinamičnih spletnih vsebin.

4.2 Avtentikacija

Pri izvedbi avtentikacije smo imeli zahtevo po obvezni vključitvi v družabno omrežje Facebook in možnost vključitve v čim več drugih družabnih omrežij. Uporabili smo Facebook Connect in OAuth.

4.2.1 Facebook Connect

Vmesnik Facebook Connect uporablja rešitev imenovano Facebook Platform. Ta preko knjižnice JavaScript omogoči vključitev spletne aplikacije v družabno omrežje Facebook. Vmesnik je bolj podrobno opisan v poglavju 2.4.3.3.

4.2.1.1 Vključitev Facebook Connect

V rešitvi smo uporabili Facebook Connect za vključitev spletne aplikacije v družabno omrežje Facebook in OAuth za vključitev ostalih družabnih omrežij, ki ta protokol podpirajo. Poglejmo si odseke programske kode JavaScript, ki prikazujejo vključitev v družabno omrežje Facebook:

- funkcija preveri, ali je uporabnik že prijavljen v Facebook in / ali v našo spletno stran preko Facebook:

```
// funkcija se kliče, ko dobi rezultat iz FB.getLoginStatus().
function statusChangeCallback(response) {
  // odgovor pove z atributom status, ali je uporabnik prijavljen
  if (response.status === 'connected') {
    testAPI();
  } else if (response.status === 'not_authorized') {
    // oseba je prijavljena v Facebook, ne pa v našo aplikacijo
    document.getElementById('status').innerHTML = 'Prosim, če se prijavite';
```

```
  } else {
    // oseba ni prijavljena v Facebook, zato ne moremo vedeti
    // ali je prijavljena v našo spletno aplikacijo
    document.getElementById('status').innerHTML = 'Prosim, če se prijavite v
Facebook';
  }
}
```

- inicializacija objekta FB in funkcija, ki se pokliče, ko uporabnik izvede prijavo:

```
function checkLoginState() {
  FB.getLoginStatus(function(response) {
    statusChangeCallback(response);
  });
}
window.fbAsyncInit = function() {
         FB.init({
           appId      : '{id_aplikacije}',
           cookie     : true,  // uporabi piškotke za shranjevanje statusa
           xfbml      : true,  // uporabi jezik za uporabo vtičnikov FB
           version    : 'v2.1' // uporabi verzijo 2.1
       });
```

- po inicializaciji objekta FB kličemo funkcijo, ki poda status prijave ročici funkcije podani v prvi točki tega primera:

```
FB.getLoginStatus(function(response) {
    statusChangeCallback(response);
  });
};
```

- asinhrono naložimo knjižnico JavaScript:

```
(function(d, s, id) {
  var js, fjs = d.getElementsByTagName(s)[0];
  if (d.getElementById(id)) return;
  js = d.createElement(s); js.id = id;
  js.src = "//connect.facebook.net/en_US/sdk.js";
  fjs.parentNode.insertBefore(js, fjs);
} (document, 'script', 'facebook-jssdk'));
```

- po uspešni prijavi se pokliče funkcija v nadaljevanju, ki prikaže uporabo vmesnika Graph API:

```
function testAPI() {
  FB.api('/me', function(response) {
```

```
        console.log('Bravo .. uspešna prijava : ' + response.name);
        document.getElementById('status').innerHTML = 'Hvala za prijavo, ' +
          response.name + '!';
      });
    }
```

- poleg kode v programskem jeziku JavaScript potrebujemo še nekaj statične vsebine – gre za definirane oznake s strani Facebook, ki se preko knjižnice programsko pretvorijo v obrazec za prijavo, kadar uporabnik še ni prijavljen v omrežje Facebook:

```
<fb:login-button scope="public_profile,email" onlogin="checkLoginState();">
</fb:login-button>
<div id="status">
</div>
```

4.2.2 OAuth

Gre za odprtokodni vmesnik za izvedbo avtentikacije in avtorizacije. Spletna aplikacija uporablja protokol za dostop vsebin na družabnih omrežjih. Protokol oziroma vmesnik je podrobno opisan v poglavju 2.4.3.1.

4.2.2.1 Vključitev OAuth

Vključitev družabnega omrežja, kot je npr. Flickr zahteva uporabo OAuth. Za lažje delo z družabnim omrežjem smo uporabili phpFlickr. Kratek primer prikazuje enostavnost vključitve na strani strežnika[7]:

```
require_once("phpFlickr.php");
$f = new phpFlickr("<api key>");
$recent = $f->photos_getRecent();
foreach ($recent['photo'] as $photo) {
  $owner = $f->people_getInfo($photo['owner']);
  echo "<a href='http://www.flickr.com/photos/" . $photo['owner'] . "/" .
       $photo['id'] . "/'>";
  echo $photo['title'];
  echo "</a> Owner: ";
  echo "<a href='http://www.flickr.com/people/" . $photo['owner'] . "/'>";
  echo $owner['username'];
  echo "</a><br>";
}
```

4.3 Tehnologija uporabljena na strežniku in odjemalcu

Spletni strežnik Apache in odjemalec uporabljata:

4.3.1 AJAX

Tehnologija AJAX preko vgrajenega objekta v brskalnik omogoča izvajanje asinhronih poizvedb in dinamično osveževanje delov spletnih strani. Z uporabo programskega jezika Javascript in obliki podatkov XML lahko zgradimo aplikacije, ki nudijo uporabniku bogato izkušnjo pri uporabi[42].

4.3.2 JSON

JSON je enostavna oblika sporočil za izmenjavo podatkov preko protokola HTTP in HTTPS. Podatki se med odjemalcem in spletnim strežnikom prenašajo v tej obliki. V naši rešitvi je to sporočilo še dodatno stisnjeno in šifrirano.

4.3.3 jQuery

Knjižnica jQuery je odprtokodna knjižnica v programskem jeziku Javascript in omogoča lažji razvoj in bogatejšo izkušnjo uporabnikov.

4.4 Podatkovni strežnik

Glavni vidik pri izbiri tehnologije podatkovnega strežnika je bila varnost in podpora strukturiranim dokumentom XML. Podatkovni strežnik Oracle 12c uporablja:

4.4.1 Podatkovni tip XMLTYPE

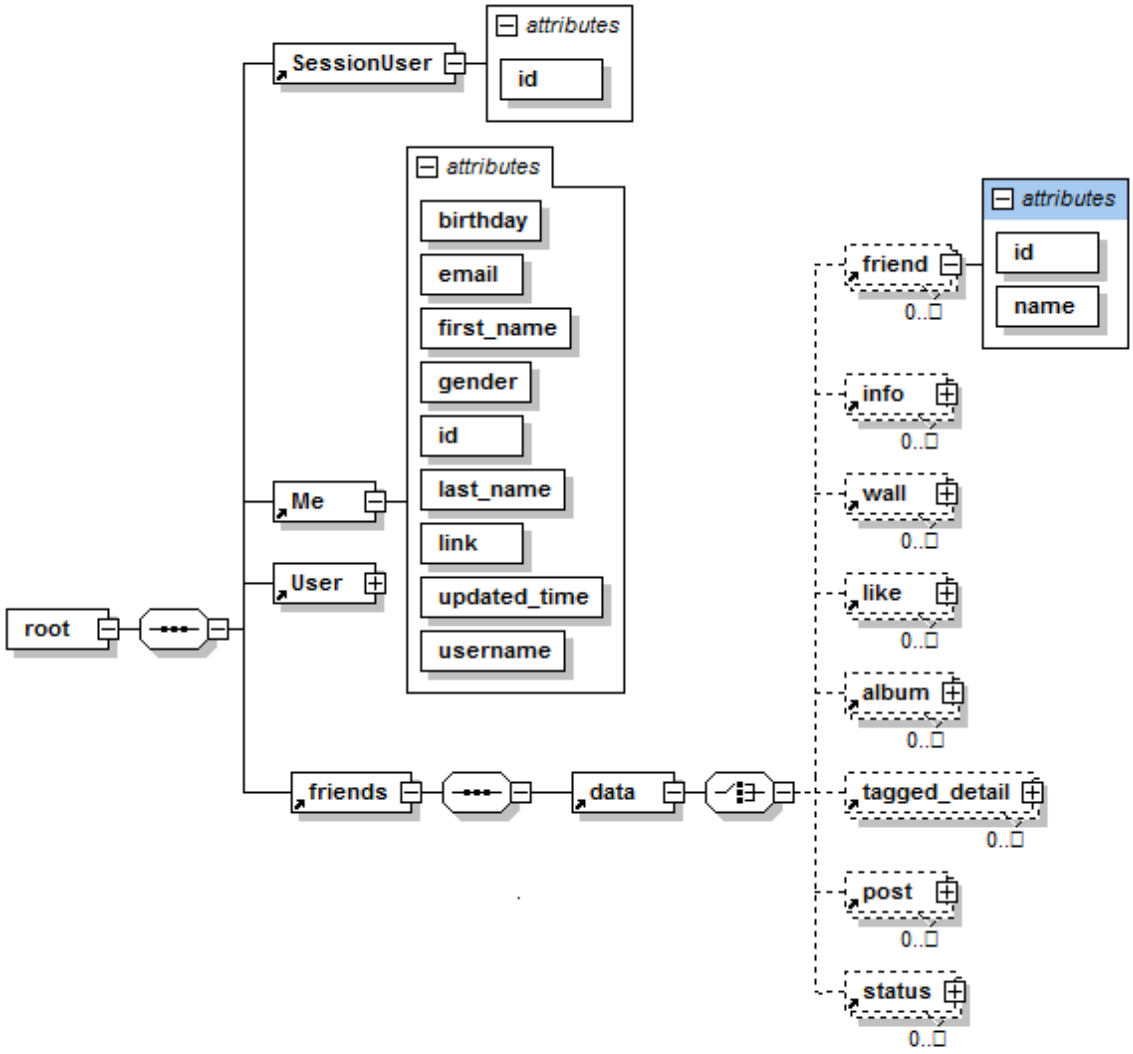
Podatkovni strežnik Oracle ima poseben tip podatka XMLTYPE. Zapis dokumenta XML v tako polje pomeni, da podatkovni strežnik samodejno preveri, če dokument ustreza osnovnim pravilom po standardu XML. V novi različici Oracle 12c je predstavljen nov tip prostora SECUREFILE za hranjene podatkov tipa XMLTYPE in omogoča napredno stiskanje ter šifriranje podatkov v obliki XML.

4.4.2 Preverjanje veljavnosti dokumentov XML glede na shemo XSD

Izbran podatkovni strežnik omogoča registracijo shem XSD v samem strežniku. S tem strežnik pridobi možnost, da izvaja preverjanje veljavnosti dokumentov XML po shemi XSD. Podatkovni strežnik izvede preverjanje pred vpisom dokumenta XML v tabelo.

Na bazi definiramo tip stolpca XMLTYPE in razvijemo shemo XSD, po kateri mora biti dokument XML veljaven. Shema XSD nam omogoča, da v dokument vpeljemo določena poslovna pravila, ki se navezujejo na vsebino polj oz. njihovo obliko. S postopkom registracije vnesemo shemo XSD v podatkovni strežnik in s tem dobimo možnost preverjanja veljavnosti dokumenta XML po shemi, ki je registrirana v podatkovnem strežniku. Pri postopku registracije strežnik izdela objektne tipe na podlagi sheme XSD. Slika 12 prikazuje del vsebine sheme XSD, katere del kode je prikazan v nadaljevanju:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema xmlns="http://www.spletno.mesto/shema.xsd"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           targetNamespace=http://www.spletno.mesto/shema.xsd
           elementFormDefault="qualified">
  <xs:element name="Me">
  <xs:complexType>
    <xs:attribute name="birthday" use="required">
      <xs:simpleType><xs:restriction base="xs:string"/></xs:simpleType>
    </xs:attribute>
    <xs:attribute name="email" use="required">
      <xs:simpleType><xs:restriction base="xs:string"/></xs:simpleType>
    </xs:attribute>
    <xs:attribute name="first_name" use="required">
      <xs:simpleType><xs:restriction base="xs:string"><xs:maxLength
value="280"/></xs:restriction>
      </xs:simpleType>
    </xs:attribute>
                    ...
  </xs:complexType>
</xs:element>
```



Slika 12: Prikaz dela sheme XSD za preverjanje veljavnosti dokumentov

Za pravilno obravnavanje datumskih polj moramo shemo opremiti z odsekom xdb. To naredimo v dveh korakih:

- opremimo glavo v razdelku *xs:schema*tako, da dodamo lastnost *xmlns:xdb="http://xmlns.oracle.com/xdb"*,
- tipom xs:datetime dodamo lastnost xdb:SQLType="TIMESTAMP WITH TIME ZONE".

Primer prikazuje omenjene transformacije na shemi XSD (odebeljeno) in registracijo sheme na bazi z blokom PLSQL:

```
BEGIN
  DBMS_XMLSCHEMA.DELETESCHEMA( 'http://www.spletno.mesto/shema.xsd',
dbms_xmlschema.delete_cascade_force );
COMMIT;

DBMS_XMLSCHEMA.REGISTERSCHEMA('http://www.spletno.mesto/shema.xsd',
    '<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns="http://www.spletno.mesto/shema.xsd"
                  xmlns:xs="http://www.w3.org/2001/XMLSchema"
                  targetNamespace="http://www.spletno.mesto/shema.xsd"
                  elementFormDefault="qualified"
xmlns:xdb="http://xmlns.oracle.com/xdb">
<xs:element name="root">
          ...
       <xs:element name="MyDate" type="xs:date" xdb:SQLType="TIMESTAMP WITH TIME
ZONE">
       ...
</xs:element>
       </xs:schema>', FALSE, TRUE, FALSE, FALSE);
  COMMIT;
END;
/
```

Na ta način se datumsko polje v bazi lahko obravnava kot datum in ne tekst ter z enostavno funkcijo podpremo številne datumske oblike, kot so:

- "1999-09-01",
- "2006-08-31T00:00:00",
- "2007-04-22T17:42:00.0Z",
- "2007-05-17T22:57:42.218+02:00",
- "2002-10-10T00:00:00+02",
- "2002-10-10-11:00".

Preverjanje veljavnosti dokumenta XML izvedemo z blokom PLSQL:

```
LX_S :=LX_XML.CREATESCHEMABASEDXML('http://www.spletno.mesto/shema.xsd');

-- izvedemo validacijo sheme
```

```
IF (LX_S.ISSCHEMAVALID('http://www.spletno.mesto/shema.xsd') <> 1) THEN

  -- v primeru da ne ustreza shemi, javi napako
 LX_S.SCHEMAVALIDATE();

END IF;
```

Na ta način zagotovimo, da je dokument XML, ki ga zapisujemo v bazo, vedno zgrajen na enak način in ustreza pravilom, določenim v shemi XSD. V shemi XSD podamo naslednje omejitve:

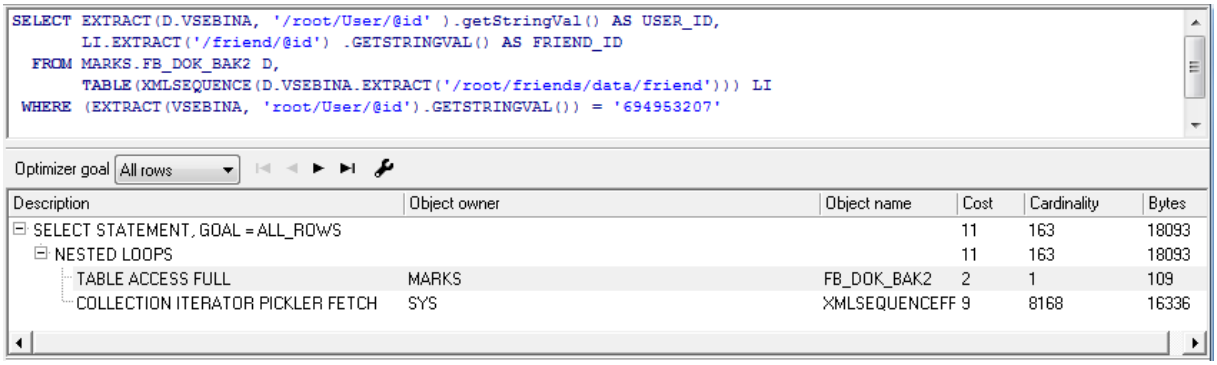
- maksimalna / minimalna dolžina polja,
- točno določena dolžina polja,
- obveznost polj,
- vzorec zapisa z uporabo posebnega jezika za iskanje podobnosti (*regular expression*).

Nad posameznimi deli dokumenta XML kreiramo funkcijski indeks, vendar moramo biti tu previdni, saj takega indeksa ne moremo uporabiti na vozlišču, ki vsebuje več vrstic. Z njim lahko optimalno iščemo po nekem ključu v dokumentu. Ko dokument najdemo, ga pridobimo celotnega iz tabele po ključu tabele in s postopkom pridobimo ostale podatke. Pri iskanju po ključu znotraj dokumenta lahko uporabimo funkcijski indeks, ki je prikazan s primerom, kjer iščemo v tabeli dokumentov uporabnika in želimo pridobiti vse njegove prijatelje. Te smo pridobivali z vsako uporabnikovo prijavo v spletno aplikacijo in sledimo lahko tudi dinamiki dodajanja novih prijateljev. Slika 12 ponazarja shemo XSD, po kateri je dokument XML veljaven in iz katerega preko stavka SQL, ko je dokument zapisan v bazi, pridobimo prijatelje:

```
SELECT EXTRACT(D.VSEBINA, '/root/User/@id' ).getStringVal() AS USER_ID,
       LI.EXTRACT('/friend/@id') .GETSTRINGVAL() AS FRIEND_ID
FROM MARKS.FB_DOK_BAK2 D,
TABLE(XMLSEQUENCE(D.VSEBINA.EXTRACT('/root/friends/data/friend'))) LI
WHERE (EXTRACT(VSEBINA, 'root/User/@id').GETSTRINGVAL()) = '694953207';
```

Brez uporabe funkcijskega indeksa bi tako iskanje pomenilo iskanje po vsebini dokumentov po celotni tabeli, ki lahko vsebuje veliko število vrstic. Slika 13 prikazuje analizo izvajanja takega stavka SQL brez funkcijskega indeksa, razvidni sta dve operaciji:

- iskanje po celotni tabeli dokumentov XML,
- iskanje po vozliščih znotraj posameznega dokumenta XML.



```
SELECT EXTRACT(D.VSEBINA, '/root/User/@id' ).getStringVal() AS USER_ID,
       LI.EXTRACT('/friend/@id') .GETSTRINGVAL() AS FRIEND_ID
  FROM MARKS.FB_DOK_BAK2 D,
       TABLE(XMLSEQUENCE(D.VSEBINA.EXTRACT('/root/friends/data/friend'))) LI
 WHERE (EXTRACT(VSEBINA, 'root/User/@id').GETSTRINGVAL()) = '694953207'
```

Optimizer goal: All rows

Description	Object owner	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL_ROWS			11	163	18093
NESTED LOOPS			11	163	18093
TABLE ACCESS FULL	MARKS	FB_DOK_BAK2	2	1	109
COLLECTION ITERATOR PICKLER FETCH	SYS	XMLSEQUENCEFF	9	8168	16336

Slika 13 : Izvedbeni plan izvajanja stavka SQL brez funkcijskega indeksa

Izvedemo po vrsti naslednje korake:

- kreiramo funkcijski indeks,

```
CREATEINDEX MARKS.X_TEST_USER ON MARKS.FB_DOK_BAK2
  (EXTRACT(VSEBINA, 'root/User/@id').GETSTRINGVAL());
```

- analiziramo in preverimo strukturo indeksa (izvede se izračun osnovne statistike indeksa),

```
ANALYZEINDEX MARKS.X_TEST_USER VALIDATESTRUCTURE;
```

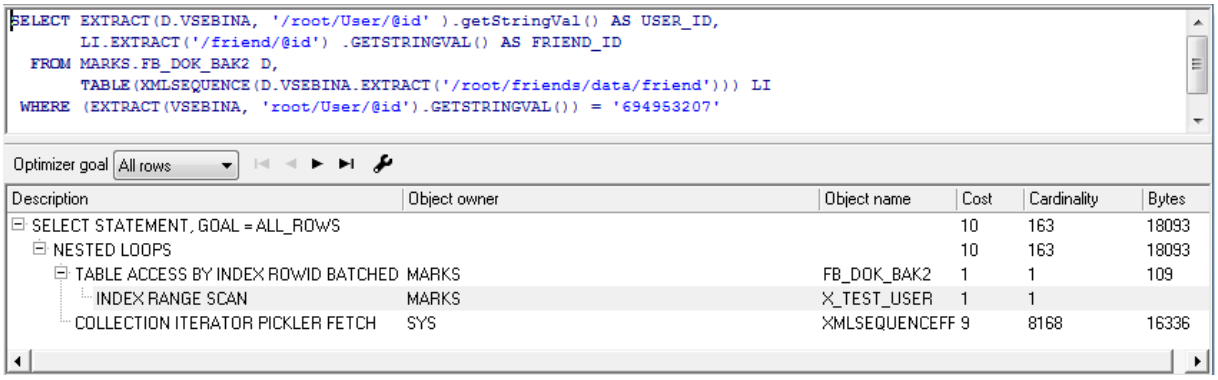
- izvedemo statistiko nad indeksom z uporabo DBMS_STATS paketa na bazi.

```
BEGIN
   DBMS_STATS.GATHER_INDEX_STATS(
     ownname => 'MARKS',
     indname => 'X_TEST_USER',
     estimate_percent => 100,
     degree => 2,
     force => TRUE);
   COMMIT;
END;
/
```

Slika 14 prikazuje izvedbeni plan izvajanja po kreiranju indeksa. Razvidno je, da se pri izvajanju:

- uporabi nov funkcijski indeks, da takoj pridemo do vrstic v tabeli, v katerih je znotraj dokumenta XML iskan uporabnik,

- iskanje po vozliščih znotraj posameznega dokumenta XML se izvaja samo na podmnožici vseh dokumentov.



Slika 14 : Izvedbeni plan izvajanja stavka SQL s funkcijskim indeksom

Rezultat iskanja je takojšen tudi pri večjem naboru vrstic v tabeli dokumentov. Bistveno boljši odziv pri iskanju dobimo, če v postopku vpisa dokumenta v tabelo dokument hkrati obdelamo s prožilci in pridobljeno oz. izluščeno vsebino iz dokumenta zapišemo vzporedno še v drugo tabelo. Nad tako tabelo kreiramo navadne indekse in iskanje je takojšnje. Posledično vpis dokumenta traja dlje, vendar lahko z optimizacijo lastnosti prostora, kamor se podatki zapisujejo, dosežemo pospešitev pri tabelah, kjer pride največ do vstavljanja podatkov (lastnost PCTFREE 0).

4.4.3 Oracle Advanced Compression and Encryption

Možnost v podatkovnem strežniku, imenovana *Oracle Advanced Compression*, omogoča stiskanje in šifriranje podatkov na mestu, kjer so fizično shranjeni (na trdem disku)[22].

Z uporabo algoritmov je tako dosežena varna komunikacija med odjemalcem in strežnikom. Če je podatkovna baza na ločenem strežniku, moramo zaščititi povezavo tudi med spletnim strežnikom in podatkovnim strežnikom. Pri podatkovni bazi Oracle to naredimo tako, da nastavimo poslušalca povezav, da deluje v varnem načinu – z uporabo protokola SSL/TLS. V primeru kraje podatkov privzete datoteke, v katerih podatkovni strežnik hrani podatke, niso šifrirane. Vpogled v tako datoteko lahko razkrije občutljive podatke. Za občutljive podatke lahko nastavimo, da se šifrirajo na nivoju stolpca. Razlikujemo še šifriranje stolpcev tipa XMLTYPE ali CLOB, v katerem je shranjen dokument XML, in šifriranje na nivoju celotnega prostora, v katerem je sklop podatkov (tipično se navezuje na lastnika podatkov in vsebuje vse podatke ter strukture tega lastnika). Na ta način imamo zaščitene tudi datoteke na podatkovnem strežniku. Uporaba šifriranja je v podatkovni bazi Oracle enostavna:

- Ročna uporaba preko baznega paketa DBMS_CRYPTO. Način šifriranja izvedemo s seštevanjem fiksnih vrednosti, ki predstavljajo tipe in možnosti šifriranja ter so podane v glavi paketa DBMS_CRYPTO. Primer:

```
DBMS_CRYPTO.ENCRYPT_AES256 +
DBMS_CRYPTO.CHAIN_CBC +
DBMS_CRYPTO.PAD_PKCS5
```

- Na nivoju enostavnega stolpca šifriranje določimo ob kreiranju tabele z argumentom ob samem stolpcu:

```
CREATE TABLE sif_table (
Sal NUMBER(6) ENCRYPT USING 'AES256'
);
```

Če tabela že obstaja, uporabimo ukaz:

```
ALTER TABLE sif_tableMODIFY (salENCRYPT USING 'AES256');
```

- Na nivoju stolpca tipa CLOB ali XMLTYPE imamo šifriranje podano pri določitvi prostora za hranjenje podatkov CLOB / XMLTYPE in ta ni nujno enak, kot je podatkovni prostor tabele. Oracle omogoča napredno šifriranje in stiskanje podatkov. Način shranjevanja z možnostjo izvajanja naprednega šifriranja in stiskanja se določi z besedo SECUREFILE:

```
CREATE TABLE sif_table (
  id         NUMBER,
  clob_data  CLOB
)
LOB(clob_data) STORE AS SECUREFILE encrypt_lob(
  ENCRYPT USING 'AES256'
COMPRESS HIGH
);
```

- Na nivoju podatkovnega prostora ob kreiranju baze oziroma ob kreiranju novega podatkovnega prostora, v katerem je shranjen nabor objektov (tipično istega lastnika, ni pa nujno). Možnost se v podatkovni bazi Oracle imenuje *Transparent Data Encryption*. Tu potrebujemo shrambo ključev, imenovano *Oracle Wallet* ali denarnica. Denarnico nastavimo tako, da v konfiguracijsko datoteko sqlnet.ora na podatkovnem strežniku dodamo nastavitev, ki je fizična pot do datoteke, kjer bo denarnica shranjena.

```
ENCRYPTION_WALLET_LOCATION=
  (SOURCE=(METHOD=FILE)(METHOD_DATA=
    (DIRECTORY=/u01/app/oracle/admin/DB10G/encryption_wallet/)))
```

- Nastavimo geslo denarnice z ukazom:

```
ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY password
/
```

- Kreiramo še podatkovni prostor, ki bo šifriran:

```
CREATE TABLESPACE ts_tde
  DATAFILE '/u01/app/oracle/oradata/ora11g/ts_tde01.dbf'
  SIZE 20m ATOEXTEND ON NEXT 5m
  EXTENT MANAGEMENT LOCAL
  SEGMENT SPACE MANAGEMENT AUTO
  ENCRYPTION USING 'AES256'
  DEFAULT STORAGE (ENCRYPT)
/
```

- Ob kreiranju tabele moramo zagotoviti, da se uporabi podatkovni prostor, ki je šifriran:

```
CREATE TABLE sif_table (
  id         NUMBER,
  clob_data  CLOB
)
TABLESPACE ts_tde
/
```

4.4.4 Oracle Text

Možnost v podatkovnem strežniku, imenovana *Oracle Text*, omogoča iskanje po strukturiranih in nestrukturiranih podatkih z izdelavo kontekstnega indeksa.Zaradi velike količine različnih podatkov, ki jih prenašamo z odjemalca na strežnik in nato v podatkovno bazo, potrebujemo mehanizem, kako te podatke zapisati v bazo in kje izvajati posamezne pretvorbe in izvedbene stavke za vpis podatkov v tabele. Zaradi zmanjšanja časovnega zamika pri izvajanju velike količine SQL stavkov je v primeru izbire podatkovne baze Oracle 12c najprimernejše izvajanje kar na podatkovni bazi. Mehanizem spletne aplikacije poskrbi, da se podatki v kosih zapišejo v bazo v obliki XML, nato pa baza poskrbi, da se vsebina izlušči in zapiše v tabele preko prožilcev. Največkrat ne pretvarjamo celotne vsebine v dokumentu XML v vrstice tabel, ampak preko podpore za delo z dokumenti XML v podatkovni bazi uporabimo vgrajene mehanizme za hitro iskanje ali pridobitev vrednosti znotraj dokumenta. Hiter odziv v poslovni logiki je zahtevan pri velikem številu uporabnikov spletne aplikacije, zato želimo podatke iz dokumenta pridobiti čim hitreje, vendar zaradi količine dokumentov XML v tabeli to navadno ni enostavno. Možnost Oracle Text nam v podatkovni bazi Oracle

omogoča kreiranje kontekstnega indeksa, kateremu preko lastnosti določimo, kakšno vsebino naj algoritem pričakuje: prosti tekst, vsebino XML ali HTML. Če shranjujemo podatke v dokumentu v obliki XML v polje tipa XMLTYPE, potem uporabimo XML_SECTION_GROUP, podobno uporabimo HTML_SECTION_GROUP za vsebino HTML, ki ga shranjujemo v polje tipa CLOB. Izvedemo korake:

- opis lastnosti za vsebino v obliki HTML:

```
BEGIN
  CTX_DDL.CREATE_SECTION_GROUP('htmgroup', 'HTML_SECTION_GROUP');
  CTX_DDL.ADD_ZONE_SECTION('htmgroup', 'heading', 'H1');
  CTX_DDL.ADD_ZONE_SECTION('htmgroup', 'title', 'TITLE');
  CTX_DDL.ADD_ZONE_SECTION('htmgroup', 'author', 'meta@author');
  CTX_DDL.ADD_ZONE_SECTION('htmgroup', 'description',
'meta@description');
  CTX_DDL.ADD_ZONE_SECTION('htmgroup', 'keywords', 'meta@keywords');
  CTX_DDL.ADD_SPECIAL_SECTION('htmgroup', 'SENTENCE');
END;
/
```

- ali opis lastnosti za vsebino v obliki XML:

```
BEGIN
  CTX_DDL_CREATE_SECTION_GROUP('xmlgroup',          'XML_SECTION_GROUP');
  CTX_DDL.ADD_ATTR_SECTION('xmlgroup',     'booktitle',     'BOOK@TITLE');
END;
/
```

- kreiranje kontekstnega indeksa:

```
CREATEINDEX MARKS.CTX_INDEX ON MARKS.DATA_URL_FRAGMENT(CLOB_DATA)
  INDEXTYPEIS CTXSYS.CONTEXT
  PARAMETERS('filter ctxsys.null_filter section group sys.htmgroup')
/
```

To je vse, kar je treba narediti, da lahko iščemo po vsebini dokumentov XML, ki so shranjeni v tabeli v stolpcu tipa XMLTYPE, ali pa po vsebini dokumentov HTML, ki so shranjeni v tabeli v stolpcu tipa CLOB. V zgornjem primeru smo za vsebino tipa HTML izbrali, da indeksiramo samo podatke, ki so v glavi, naslovu ali v sklopu meta podatkov (avtor, opis in ključne besede).

Iskanje po vsebini izvajamo s posebno obliko stavkov SQL. Primer enostavne uporabe prikazuje, kako izvedemo iskanje besede *Google* v dokumentih, shranjenih v podatkovni bazi:

```
SELECT D.RANK,
       D.ID,
       D.URL
FROM (SELECT SCORE(1) AS RANK, D.ID, D.URL
        FROM MARKS.DATA_URL_FRAGMENT D
       WHERE CONTAINS(CLOB_DATA, 'Google', 1) >0
     ) D
ORDERBY RANK DESC;
```

V rezultatu lahko prikažemo tudi oceno zadetka. Naprednejša uporaba prikazuje, na kakšen način bo Oracle pridobil rezultate v primeru uporabe več besed (besede, povezane med sabo z veznikom AND, vejico ali presledkom):

```
SELECT SCORE(1) AS RANK,
       D.ID,
       D.URL
  FROM MARKS.DATA_URL_FRAGMENT D
 WHERE CONTAINS(D.CLOB_DATA,
'<QUERY>
   <TEXTQUERY GRAMMAR="CONTEXT">Google education
     <PROGRESSION>
       <SEQ><REWRITE>TRANSFORM((TOKENS, "{", "}", " "))</REWRITE></SEQ>
       <SEQ><REWRITE>TRANSFORM((TOKENS, "{", "}", "AND"))</REWRITE></SEQ>
       <SEQ><REWRITE>TRANSFORM((TOKENS, "{", "}", ","))</REWRITE></SEQ>
     </PROGRESSION>
   </TEXTQUERY>
   <SCORE DATATYPE="INTEGER" ALGORITHM="COUNT"/>
</QUERY>', 1 ) > 0
ORDER BYRANK DESC;
```

4.5 Izmenjava podatkov

Povezava med odjemalcem in ponudniki družabnih storitev poteka po protokolu SSL/TLS in HTTPS. Kaj pa takrat, ko SSL protokola ne moremo uporabiti? Preverili in uporabili smo podporo za šifriranje in stiskanje na strani odjemalca in strežnika.

4.5.1 Podpora šifriranju na strani odjemalca

Programska koda za podporo šifriranja AES na strani odjemlca zahteva naslednje knjižnice v programskem jeziku JavaScript za podporo šifriranja in transformacij:

- AES Cipher – Advanced Encryption Standard (FIPS PUB 197[24] – osnova za podporo šifriranja po standardu AES,

- AES Counter-mode (SP 800-38A[25] za podporo šifriranja v blokovnem načinu delovanja,
- kodiranje Base64 (RFC 4648[27] za podporo prenosa podatkov, ki so omejeni na nabor znakov US-ASCII,
- kodiranje UTF-8 za podporo transformacije med Unicode in UTF-8 naborom znakov.

Za podporo asinhronega načina delovanja je uporabljena knjižnica jQuery z naslednjimi vtičniki, ki razširijo zmožnost jQuery knjižnice z novimi ukazi:

- jQuery 1.4.2 – osnovna knjižnica za podporo jQuery[45],
- jQuery Easing Plugin v1.3 in jQuery Easing Compatibility v1[43],
- jQuery Timer plugin v0.1– metode za časovne operacije[41],
- jQuery Ajax Queue Plugin – metode za delo z vrsto[13],
- jQuery JSONP Core Plugin 2.1.3 – metode za podporo JSON protokolu[4].

Za izvedbo asinhronih klicev na spletno mesto se uporablja objekt ActiveX v brskalniku Internet Explorer (če je na voljo), sicer se uporabi objekt XMLHttpRequest. Z uporabo knjižnic jQuery je uporaba enostavnejša in tudi primer v nadaljevanju je prikazan z uporabo te knjižnice. Spletna aplikacija izvaja klice spletne strani, ki je povezana s podatkovno bazo in po potrebi shrani podatke, jih obdela ter vrne odgovor. Koda na strani strežnika odgovor ali navodila za izvedbo opravila šifrira, stisne in izvede zahtevane transformacije, da ne pride do izgube podatkov ter posreduje odgovor. Odgovor je lahko sporočilo v obliki HTML ali pa struktura, serializirana (pretvorjena) v tekst (JSON), kar je najpogostejša uporaba zaradi kasnejše enostavnejše obdelave podatkov na strani odjemalca. Odjemalec namreč strukturo deserializira (pretvori iz teksta v strukturo s polji oz. atributi) in izvede kodo v globalni kontekst. S tem se na strani odjemalca pojavi enaka struktura, kot je bila na strežniku, brez potrebe po definiranju strukture podatkov (koda se ovrednoti oziroma izvede). Za kratka sporočila lahko uporabimo metodo GET, kot prikazuje primer, metoda POST se uporablja za izmenjavo večje količine podatkov po obsegu. Z definiranjem stolpca tipa CLOB na podatkovni bazi Oracle je omejitev velikosti 4GB (štiri miljarde 8-bitnih besed), kar navadno ne predstavlja omejitev, pač pa največkrat pride do omejitev ali na strani spletnega strežnika ali pa na zaščitni steni na usmerjevalniku. Primer z uporabo metode GET prikazuje pošiljanje šifriranega sporočila, sprejem odgovora in dešifriranje odgovora z uporabo stiskanja:

```
var mdata = "Sporočilo ki ga pošiljamo šifriramo.";
// Nastavimo osnovno povezavo
var nurl = "http://www.spletnomesto.si/ajax/request.php?";
var mkey = "01234567890123456789012345";
// Stiskanje in base 64 transformacija
var mcomp = mBase64.toBase64(RawDeflate.deflate(mBase64.utob(mdata)));
// šifrirano "plain" tekst
var cdata = AesCtr.encrypt(mcomp, mkey, 256);
// šifrirano sporočilo podamo kot GET parameter
if( cdata != null )
    nurl = nurl + "d=" + cdata;

// izvedemo asinhron klic
  lsRespond = jQuery.ajax({
        url: nurl,
        global: true,
        type: "GET",
        cache: false,
        dataType: "html",

// obravnavamo odgovor
        success: function(msg_respond){
var mresp = "";
var mtmp1 = AesCtr.decrypt(msg_respond, key, 256);
       var mtmp2 = MAES.Util.Base64.decodeAsArray(mtmp1);
       var mtmp3 = new MAES.Util.Unzip(mtmp2);

       if( mtmp3 != undefined ) {
       var mtmp4 = mtmp3.unzip();
       if( mtmp4 != undefined && mtmp4[0] != undefined ) {
       var mtmp5 = mtmp4[0][0];
              mresp = unescape(mtmp5);
            }
          }
          gcode = MultiLineTextToSingleLineText(mresp);
globalEval(gcode);
        }
     }
  ).responseText;
}
```

Ključ za šifriranje je v primeru podan fiksno, v resnici se vsebina ključa dinamično spreminja z dodatnim algoritmom.

4.5.2 Podpora šifriranju na strani strežnika

Izvajanje stiskanja je v programskem jeziku PHP že podprto z ukazoma gzinflate in gzcompress. Za šifriranje uporabimo knjižnico AES, izdelano v programskem jeziku PHP[47]. Uporaba je podobna tisti v programskem jeziku Javascript:

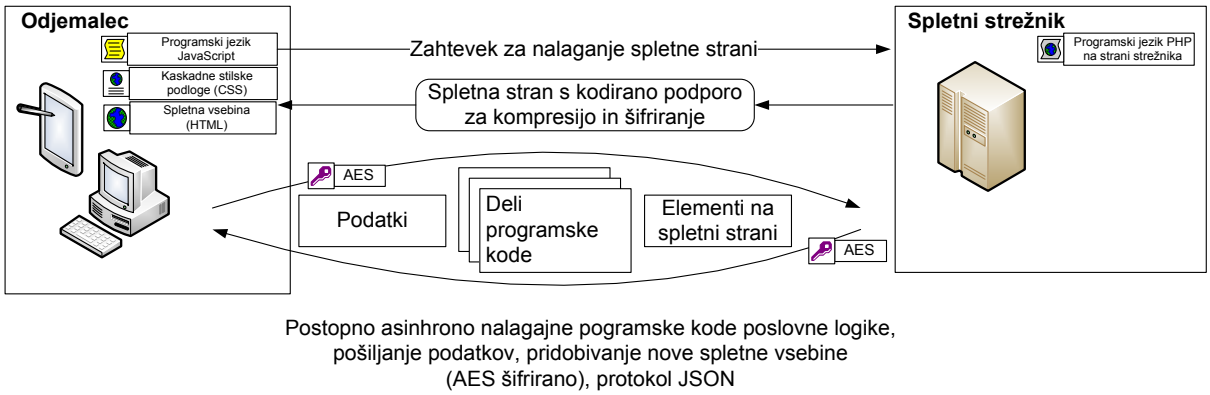
- Funkcija *AESEncryptCtr* vrne šifrirano sporočilo podano v vhodnem argumentu *$sporocilo:*

- *function AESEncryptCtr*($sporocilo, $kljuc, $st_bitov)

- Funkcija *AESDecryptCtr* vrne dešifrirano sporočilo podano v vhodnem argumentu *$en_sporocilo:*

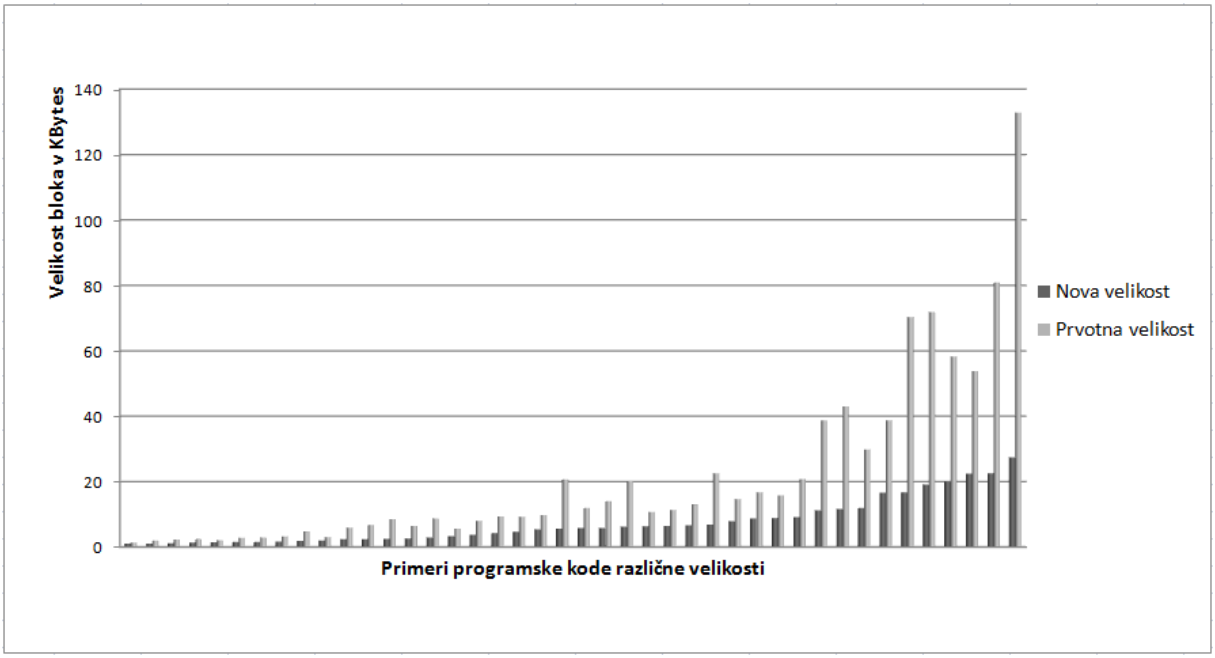
- *function AESDecryptCtr*($en_sporocilo, $kljuc, $st_bitov)

Vgradnja na strani strežnika je torej dokaj enostavna.Slika 15 prikazuje, kako se najprej naloži sistem za podporo stiskanja in šifriranja, sledi asinhrona izmenjava podatkov na varen način.



Slika 15: Prikaz varne izmenjave v primeru, da ni možna uporaba protokola SSL/TLS

Rešitev ponuja še eno prednost. Gre za velikost podatkov, ki jih prenašamo po mreži. Slika 16 prikazuje primerjavo med prvotno velikostjo in stisnjeno za različne vsebine podatkov.



- Slika 16: Prvotna velikost v primerjavi s transformirano velikostjo v KB

4.6 Orodja uporabljena pri razvoju

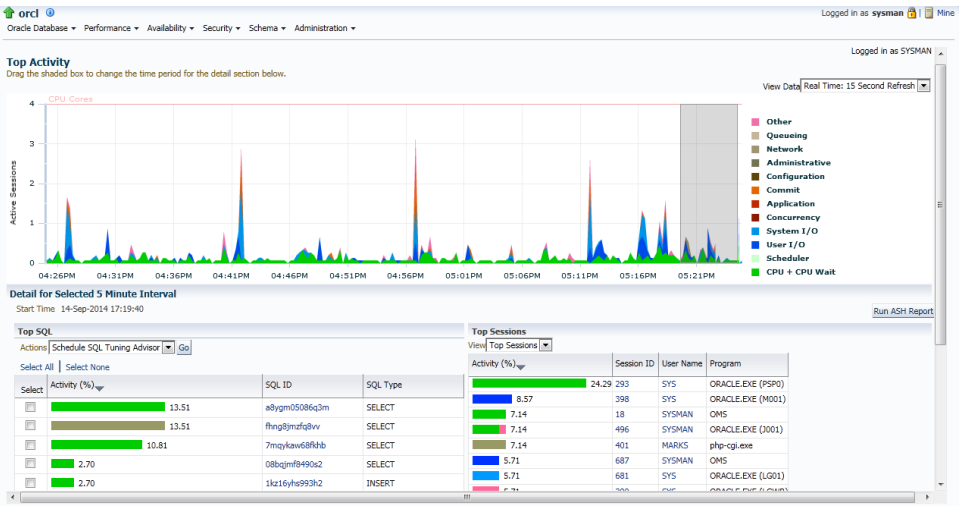
Uporabili smo naslednja orodja:

- Urejevalnik programske kode,
- Oracle SQL Developer Data Modeler,
- Oracle SQL Developer,
- Oracle 12c Database Enterprise Edition,
- Oracle 12c Cloud Control.

Oracle SQL Developer Data Modeler omogoča izgradnjo logičnih in relacijskih podatkovnih modelov ter takojšnjo izvedbo oziroma postavitev modela na podatkovno bazo. Oracle SQL Developer je orodje za izvajanje stavkov SQL. Orodje Oracle 12c Cloud Control je nadzorni sistem za vzdrževanje in pregled izvajanja podatkovne baze ter spletnih strežnikov. Na nivoju podatkovne baze zagotavlja vpogled v izvajanje posameznega stavka SQL. Z vgrajenimi orodji nam predlaga postopke za optimizacijo v primeru slabega izvajanja, kot so:

- kreiranje dodatnih indeksov za bolj optimalno izvajanje,
- spremembe, povezane s statistiko nad tabelami in indeksi,
- prepis stavka SQL za bolj optimalno izvajanje.

Slika 17 prikazuje stran za pregled najaktivnejših stavkov SQL.



Slika 17: Nadzorni sistem Oracle 12c Cloud Control

Za spletni strežnik je bil uporabljen spletni strežnik Apache, ki je že vgrajen v Oracle 12c Cloud Control. Ta je brez podpore PHP in brez certifikata, ki je zaupanja vreden, zato so bile potrebne ročne spremembe v nastavitvah:

- vgradnja podpore PHP v konfiguracijske datoteke,

vnos zaupanja vrednega strežniškega certifikata v Oracle Wallet. Oracle 12c Cloud Control namreč nima standardnega vmesnika za podporo protokola SSL/TLS, pač pa uporablja OSSL (Oracle SSL), ki se razlikuje od standardnega vmesnika po tem, da so certifikati shranjeni v denarnici in ne kot datoteke na strežniku. Denarnica pa je dodatno šifrirana z geslom.

Poglavje 5 Sklepne ugotovitve

Ljudje se zanašamo na čutila in na podlagi čutil določimo identiteto osebe, s katero komuniciramo. Tudi v elektronskem svetu moramo pogosto enolično določiti identiteto uporabnika. Sistemov, kjer uporabljamo e-identiteto, je vedno več in upravljanje z identitetami lahko predstavlja velik izziv, zlasti če so sistemi med seboj povezani. Lahko imamo ločene podatkovne strežnike z atributi identitet, kar otežuje upravljanje z identitetami. Učinkovito upravljanje z uporabniškimi identitetami predstavlja danes v organizacijah pomemben proces. Organizacije se srečujejo z mnogimi izzivi – na eni strani morajo nuditi dostop do informacij uporabnikom, hkrati pa zagotoviti varnost in skladnost s predpisi in regulativo.V idealnem svetu bi si želeli centralni sistem upravljanja, vendar to ni vedno mogoče. V resničnem svetu uporabnik potrebuje dostop do več informacijskih sistemov hkrati. Družabna omrežja pogosto pomenijo za posameznika uvedbo nove identitete. Nekatera družabna omrežja in organizacije dovoljujejo posamezniku celo uporabo več digitalnih identitet hkrati.

Podjetja so v sklopu modernizacije svojih sistemov začela v delovnih procesih uporabljati družabna omrežja kot so Facebook, Twitter, Flickr ipd. Lahko bi rekli, da modernizacija sistemov vpeljuje družabna omrežja. Poleg storitev, ki jih družabna omrežja ponujajo, igrajo pomembno vlogo identitete in družabna omrežja so danes pogosto tudi ponudniki upravljanja z identitetami. Uporaba spletnih aplikacij, ki bi vsaka zase zahtevala od uporabnika izvedbo avtentikacije, bi bila neprijazna in bi uporabniku povzročila negativno izkušnjo pri uporabi. Pri spletnih aplikacijah za širši krog je lahko zagotavljanje same identitete zadostno in lahko celotno upravljanje z identitetami prepustimo ponudniku družabnih storitev. V taki aplikaciji bi uporabili protokol OpenID. V spletne aplikacije, ki so vpete v poslovne procese, pa je natančna identiteta uporabnika z dodatnimi atributi pomembna. Taka protokola sta Facebook Connect in OAuth. Protokol OAuth je široko uporabljen pri različnih ponudnikih družabnih omrežij, hkrati je različica 2 tudi uporabljena pri vmesniku Facebook Graph. Prva različica OAuth se ne spreminja več in danes imamo z vključitvijo knjižnic na strani strežnika in odjemalca možnost izdelave spletnih aplikacij, ki vključujejo storitve številnih ponudnikov. Ker je seznam dolg, naštejmo nekatere izmed največjih ponudnikov storitev: Amazon, AOL, Dropbox, Evernote, Facebook Graph API, Flickr, GitHub, Google, Instagram, LinkedIn, Microsoft (Hotmail, Windows Live, Messenger, Xbox), MySpace, PayPal, Tumblr, Twitter,

Yahoo! in še bi lahko naštevali. Pri razvoju spletnih aplikacij se zdijo protokoli, ki ponujajo pridobitev večjega nabora atributov pri izvedbi avtentikacije bolj primerni, saj ponudniku spletne aplikacije omogočajo pridobitev informacij o uporabniku, katere lahko uporabimo za zagotavljanje boljše storitve. Novejše različice protokolov, kot sta npr. OpenID in OAuth, se še vedno spreminjajo, zato je potrebna previdnost pri uporabi obstoječih knjižnic. Hitro se lahko zgodi, da knjižnica ne bo več podpirala uradnega protokola, kar bo povzročilo zastoj pri razvoju ali celo izpad dohodka. Pridobitev elektronskega naslova uporabnika je pomembno, saj lahko ostane edini način vzpostavitve stika z uporabnikom, ko pride do nezaželjenih in neplaniranih sprememb vmesnikov. V zadnjih letih smo lahko videli številne spremembe pri uporabi protokolov velikih organizacij, kot je podjetje Google. Podjetje ukinja številne načine avtentikacije oziroma uporabo protokolov, kot so OAuth 1.0, OAuth 2.0, OpenID 2.0 in uvaja novo različico protokola, ki deluje v okviru družabnega omrežja Google+. Določene različice protokolov, kot je npr. OAuth 1.0c, se ne spreminjajo več, vendar ne nudijo dovolj velike zaščite v primeru prisluškovanja povezavi in uporabi nezaščitene povezave. Z uporabo varnih protokolov na nivoju komunikacije, kot je SSL/TLS, se izognemo tem težavam. Pri izdelavi spletne aplikacije za širok krog ljudi, ne pričakujemo tehnično podkovanih strokovnjakov in v spletni aplikaciji želimo zagotoviti osnovno varnost kot je onemogočen vpogled v občutljive podatke. Kadar uporaba zaščitenih protokolov na nivoju komunikacije ni možna, lahko podatke pri prenosu šifriramo z obstoječimi algoritmi in s tem zagotovimo zaščito pred vpogledi. Velika množica identitet in shranjeni atributi v strežniku za namen zagotavljanja boljše storitve lahko predstavlja problem. Orodja napadalcev so vedno močnejša in v zadnjem času smo lahko zabeležili več kraj velikega števila identitet. Če je v naboru identitet tudi identiteta podatkovnega strežnika, so izpostavljeni vsi podatki na strežniku. Varnost na nivoju shranjevanja podatkov je zato pomembna, še posebej ko gre za shranjevanje občutljivih podatkov, kot so podatki identitet.

V diplomi sem pregledal načine avtentikacije od klasičnih do sodobnih, posvetil sem se zlasti avtentikaciji prek družabnih omrežij in na primeru pokazal, kako se jih uporablja. Ugotovil sem, da uporaba storitev zunanjega ponudnika za izvedbo avtentikacije lahko zadostuje za izvedbo avtentikacije na spletni strani, ki je vpeta v poslovni proces podjetja. Pri tem je naša spletna aplikacija izpostavljena, saj je dostopnost naše aplikacije odvisna od zunanjih storitev. Prednost prinaša poenostavljeno upravljanje z identitetami in vključitev številnih storitev, ki jih družabna omrežja nudijo in povečujejo povezljivost med organizacijami znotraj podjetja.

Zaključek pri izdelavi diplomske naloge je, da moramo danes dobro preučiti sistem in ugotoviti ali potrebujemo centralni sistem upravljanja z identitetami. Preučiti moramo varnost spletnih poti in protokolov, ki se uporabljajo na nivoju komunikacije. V kolikor sama komunikacija ne omogoča varne povezave imamo možnost uporabiti napredne mehanizme

šifriranja, ki to deloma zagotavljajo. Ko gledamo spletno aplikacijo s podporo storitvam družabnim omrežjem, lahko prepustimo upravljanje z identitetami zunanjemu ponudniku. Pri tem moramo uporabiti protokole, ki to omogočajo na varen način. Protokoli se spreminjajo, zato je danes za dostop do storitev spletnih omrežij najboljša kombinacija uporabe varne poti po protokolu SSL/TLS in uporaba protokolov, ki se ne spreminjajo. Zagotovo pa bomo v bodoče vsaj za varnostno manj občutljive aplikacije videli še velik razmah na področju avtentikacije v družabnih omrežjih.

Literatura

[1] Computing & Communications Services. (2014). *SSO Benefits*. Pridobljeno iz CCS: https://www.uoguelph.ca/ccs/security/internet/single-sign-sso/benefits

[2] Agito d.o.o. (20. Februar 2014). Upravljanje identitet v podjetju. *Finance* , str. 19.

[3] Agito d.o.o. (April 2013). Varnost in upravljanje z identitetami. *Računalniške novice* , str. 34.

[4] Aubourg, J. (Avgust 2012). *jQuery-JSONP · GitHub*. Pridobljeno iz GitHub · Build software better, together: https://github.com/jaubourg/jquery-jsonp

[5] Barkley, J. (1997). *Comparing simple role based access control models and access control lists.* Pridobljeno iz Proceedings of the second ACM workshop on Role-based access control: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.107.6366, str. 127-132

[6] Ciglarič, M., Krevl, A.,Pančur, M. (2008). *Strategija upravljanja z digitalnimi identitetami na Univerzi v Ljubljani, različica 1.0.* Ljubljana: Univerza v Ljubljani.

[7] Coulter Dan. (brez datuma). *A PHP wrapper for the Flickr API*. Prevzeto 2014 iz GitHub: https://github.com/dan-coulter/phpflickr

[8] Cugley, D. (Avgust 2007). *OpenID versus Single-Sign-On Server - Alleged Articles*. Prevzeto September 2014 iz Alleged Literature: http://alleged.org.uk/pdc/2007/08/13.html

[9] Ecma International. (2011). *ECMAScript Language Specification - ECMA-262 Edition 5_1*. Pridobljeno iz Welcome to Ecma International: http://www.ecma-international.org/ecma-262/5.1/

[10] Eržen, M. (November 2012). Odprta avtentikacija (Open authentication). Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, Ljubljana.

[11] Ferraiolo, D.F. in Kuhn, D.R. (Oktober 1992). *Role-Based Access Control.* Pridobljeno iz 15th National Computer Security Conference: http://csrc.nist.gov/groups/SNS/rbac/documents/ferraiolo-kuhn-92.pdf, str. 554–563

[12] Ferraiolo, D.F., Kuhn, D.R., in Sandhu, R. (November-December 2007). *RBAC Standard Rationale: comments on a Critique of the ANSI Standard on Role-Based Access Control.* Pridobljeno iz IEEE Security & Privacy (IEEE Press): http://csrc.nist.gov/groups/SNS/rbac/documents/ferraiolo-kuhn-sandhu-07.pdf, Vol. 5/6, str. 51-53

[13] Frang, C. (Januar 2013). *jQuery Ajax Queue jQuery Plugin Registry*. Pridobljeno iz jQuery Plugin Registry: http://plugins.jquery.com/ajaxQueue/

[14] Hitachi ID Systems, Inc. (2014). *Beyond roles - A Practical Approach to Enterprise User Provisioning.* Pridobljeno iz Identity Management and Access Governance Solutions from Hitachi ID Systems: http://hitachi-id.com/identity-manager/docs/beyond-roles.pdf

[15] Hribar, K. (September 2014). *Novi tipi omrežnih napadov in njihovo preprečevanje (Analysis and mitigation of new network attack types)* . Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, Ljubljana.

[16] IEEE. (Oktober 2004). *Standard Specifications For Public-Key Cryptography*. Pridobljeno iz The IEEE P1363 Home Page: http://grouper.ieee.org/groups/1363/

[17] Internet Engineering Task Force (IETF), Barth, A. (April 2011). *RFC 6265 - HTTP State Management Mechanism*. Pridobljeno iz Internet FAQ Archives: http://www.faqs.org/rfcs/rfc6265.html

[18] Internet Engineering Task Force (IETF), Hammer-Lahav, E. (April 2010). *RFC 5849*. Pridobljeno iz The OAuth 1.0 Protocol: http://tools.ietf.org/html/rfc5849

[19] Kragelj, V. (September 2014). *Pregled in analiza tehnologij za serializacijo objektov (Review and analysis of object serialization techniques)* . Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, Ljubljana.

[20] Marand d.o.o. (Junij 2009). *Marand_si - Upravljanje z identitetami*. Pridobljeno iz Marand_si: http://www.marand.si/resitve/upravljanje-identitet/

[21] Mena, J. (2011). *Machine Learning Forensics for Law Enforcement, Security, and Intelligence.* Boca Raton, Florida: CRC Press (Taylor & Francis Group).

[22] Nanda, A. (Julij/Avgust 2008). TECHNOLOGY: Advanced Compression - Compress to Impress. *Oracle Magazine* .

[23] Nanda, A. (Januar/Februar 2009). TECHNOLOGY: Security - Encrypting Tablespaces. *Oracle Magazine* .

[24] National Institute of Standards and Technology (NIST) . (26. November 2001). *Announcing the Advanced Encryption Standard (AES)*. Pridobljeno iz NIST Computer Security Publications - FIPS (Federal Information Processing Standards): http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[25] National Institute of Standards and Technology (NIST). (December 2001). *Recommendation for Block Cipher Modes of Operation - Methods and Techniques.* Pridobljeno iz NIST Computer Security Publications - FIPS (Federal Information Processing Standards): http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

[26] Network Working Group. (1996). *RFC 2045*. Pridobljeno iz RFC-Base_org the central repository of all Requests For Comments RFC documents: http://www.rfc-base.org/rfc-2045.html

[27] Network Working Group. (Oktober 2006). *RFC 4648 - The Base16, Base32, and Base64 Data Encodings*. Pridobljeno iz IETF Tools: http://tools.ietf.org/html/rfc4648

[28] Network Working Group, Franks J., Hallam-Baker P., Hostetler J., Lawrence S., Leach P., Luotonen A., Stewart L. (Junij 1999). *HTTP Authentication: Basic and Digest Access Authentication*. Pridobljeno iz IETF Tools: http://www.ietf.org/rfc/rfc2617.txt

[29] Network Working Group, Franks J., Hallam-Baker P., Hostetler J., Leach P., Luotonen A., Stewart L., Sink E. (Januar 1997). *An Extension to HTTP : Digest Access Authentication*. Pridobljeno iz IETF Tools: http://tools.ietf.org/html/rfc2069

[30] Network Working Group, Krawczyk H., Bellare M., Canetti R. (Februar 1997). *HMAC: Keyed-Hashing for Message Authentication*. Pridobljeno iz IETF Tools: http://www.ietf.org/rfc/rfc2104.txt

[31] Network Working Group, Kristol D., Montulli L. (Februar 1997). *HTTP State Management Mechanism*. Pridobljeno iz The Internet Engineering Task Force (IETF): https://www.ietf.org/rfc/rfc2109.txt

[32] Network Working Group, Shirey, R. (Avgust 2007). *Internet Security Glossary, Version 2*. Pridobljeno iz IETF Tools: http://tools.ietf.org/html/rfc4949

[33] Network Working Group, Wu, T. (September 2000). *The SRP Authentication and Key Exchange System*. Pridobljeno iz RFC 2945: http://tools.ietf.org/html/rfc2945

[34] OASIS (Advancing open standards for the information society). (1993). *Online comunity for the Security Assertion Markup Language*. Pridobljeno iz SAML: http://saml.xml.org/

[35] O'Connor, A.C. in Loomis, R.J. (December 2010). *Economic Analysis of Role-Based Access Control.* Pridobljeno iz Research Triangle Institute: http://csrc.nist.gov/groups/SNS/rbac/documents/20101219_RBAC2_Final_Report.pdf

[36] Pfitzmann, A., & Hansen, M. (10. Avgust 2010). *Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management*. Prevzeto September 2014 iz A terminology for talking about privacy by data minimization (Version v0.34): http://dud.inf.tu-dresden.de/Anon_Terminology.shtml

[37] Podgornik, D. (Maj 2011). *Uporaba konceptov spleta druge generacije pri izgradnji spletnih aplikacij (Use of Web 2.0 concepts in development of web applications)* . Diplomsko delo, Fakulteta za računalništvo in informatiko, Univerza v Ljubljani, Ljubljana.

[38] Reffell, J. (Marec 2011). *Oauth, OpenID, Facebook Connect: Authentication Design Best Practices*. Prevzeto September 2014 iz Upload, Share, and Discover Content on SlideShare: http://www.slideshare.net/fullscreen/jreffell/oauth-openid-facebook-connect-authentication-design-best-practices

[39] Sandhu, R., Coyne, E.J., Feinstein, H.L. in Youman, C.E. (Avgust 1996). *Role-Based Access Control Models.* Pridobljeno iz IEEE Computer (IEEE Press): http://csrc.nist.gov/rbac/sandhu96.pdf, Vol. 29/2, str. 38–47

[40] Sandhu, R., Ferraiolo, D.F. in Kuhn, D.R. (Julij 2000). *The NIST Model for Role-Based Access Control: Toward a Unified Standard.* Pridobljeno iz 5th ACM

Workshop Role-Based Access Control: http://csrc.nist.gov/rbac/sandhu-ferraiolo-kuhn-00.pdf, str. 47-63

[41] Schmidt, M. (Januar 2012). *Projects jQuery timer*. Pridobljeno iz Matt ptr: http://mattptr.net/pages/projects.html

[42] Schwerin, R. (September/Oktober 2006). Getting Rich with Ajax - Build rich internet applications with Asynchronous JavaScript and XML. *Oracle Magazine* .

[43] Smith, G. M. (November 2007). *jQuery Easing Plugin*. Pridobljeno iz Foundation: http://gsgd.co.uk/sandbox/jquery/easing/

[44] Smith, J. (Maj/Junij 2014). TECHNOLOGY: SQL Developer, Oracle SQL Developer Data Modeler - Document Entities. *Oracle Magazine* .

[45] The jQuery Team. (2014). *jQuery*. Pridobljeno iz jQuery: http://jquery.com/

[46] Tilkov, S. (02. Julij 2008). *REST Anti-Patterns*. Prevzeto 2014 iz InfoQ Software Development News, Videos & Books: http://www.infoq.com/articles/rest-anti-patterns

[47] Veness, C. (2005). *PHP Implementation of AES Advanced Encryption Standard in Counter Mode*. Pridobljeno iz Movable Type — Information Design & Management: http://www.movable-type.co.uk/scripts/aes-php.html

[48] Videc, Ž. (Junij 2012). Upravljanje z uporabniki. Diplomsko delo, Fakulteta za elektrotehniko, računalništvo in informatiko, Univerza v Mariboru, Maribor.

[49] Wikipedia, the free encyclopedia. (September 2014). *Identity management*. Pridobljeno iz Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/Identity_management

[50] Wikipedia, the free encyclopedia. (Oktober 2014). *JSON*. Pridobljeno iz Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/JSON

[51] Wikipedia, the free encyclopedia. (05. Oktober 2014). *List of LDAP software*. Pridobljeno iz Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/List_of_LDAP_software

[52] Wikipedia, the free encyclopedia. (Oktober 2014). *List of single sign-on implementations*. Pridobljeno iz Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/List_of_single_sign-on_implementations